

WILEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

CREATION OF A TRANSPORTABLE INTERACTIVE USER
INTERFACE FOR IMPROVED AAA SIMULATION
COMPUTER PROGRAM (P001)

by

Eric R. Johns

June 1982

Thesis Advisor:

R. E. Ball

Approved for public release; distribution unlimited.

T204506

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Creation of a Transportable Interactive User Interface for Improved AAA Simulation Computer Program (P001)		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1982
7. AUTHOR(s) Eric R. Johns		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		6. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE June, 1982
		13. NUMBER OF PAGES 171
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Aircraft survivability, interactive graphics.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Air Force Armament Laboratory AAA Simulation Computer Program and MICE II have been adapted for batch processing use on the Naval Postgraduate School IBM 3033 computer. To ease data entry and reduce errors, a preprocessor program (PIP) was written at the school. The modifications necessary to convert PIP from a batch program to an interactive one are described herein.		

This conversion to an interactive program has two goals: graphics capability and portability. The revised versions are designed in a manner that allows users to modify routines to meet their particular hardware and software. The graphics capability implements PIP using TEKTRONIX 4010 family hardware and PLOT-10 software. Derivative versions implement the program using the IBM 3277 graphics system and using strictly keyboard versions. In addition to PIP, a program for graphic design of scenario maps was developed.

Approved for public release; distribution unlimited.

Creation of a Transportable Interactive User Interface
for Improved AAA Simulation Computer Program (P001)

by

Eric R. Johns
Lieutenant, United States Navy
B.S.S.E., United States Naval Academy, 1975

Submitted in partial fulfillment of the
requirement for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the
NAVAL POSTGRADUATE SCHOOL

June 1982

ABSTRACT

The Air Force Armament Laboratory AAA Simulation Computer Program and MICE II have been adapted for batch processing use on the Naval Postgraduate School IBM 3033 computer. To ease data entry and reduce errors, a preprocessor program (PIP) was written at the school. The modifications necessary to convert PIP from a batch program to an interactive one are described herein.

This conversion to an interactive program has two goals: graphics capability and portability. The revised versions are designed in a manner that allows users to modify routines to meet their particular hardware and software. The graphics capability implements PIP using TEKTRONIX 4010 family hardware and PLOT-10 software. Derivative versions implement the program using the IBM 3277 graphics system and using strictly keyboard versions. In addition to PIP, a program for graphic design of scenario maps was developed.

TABLE OF CONTENTS

I.	INTRODUCTION	13
II.	APPROACH	17
	A. GOALS	17
	B. METHODOLOGY	18
III.	FUNCTIONAL DESCRIPTION	20
	A. DATA SETS	20
	1. Internal Sets	20
	2. External Sets	21
	B. MODULES	21
	1. MAIN	21
	2. VALSET	22
	3. ERRCHK	22
	4. PTHPLT	23
	5. SPOT	23
	6. SCENE	23
	7. WIN	23
	8. XYZIN	24
	9. GUNLOC	24
	10. BEGIN	24
	11. GUNCHK	24
	12. PRESET	24
	13. ELFIN	25

14.	AIMPT	25
15.	CONCHG	25
16.	ERRMK	25
17.	SAMCHK	25
IV.	FLIGHT DYNAMICS	26
V.	GRAPHICS HARDWARE AND SOFTWARE	33
VI.	DETAILED DESCRIPTION	39
A.	INTERNAL DATA SETS	40
1.	Screen Window Defined	40
a.	IBAUD	40
b.	MINY, MAXY	40
c.	MINX, MAXX	41
d.	MIN1, MAX1	41
2.	Target Definitions	41
a.	TARGX	41
b.	TARGY	42
3.	Option List	42
a.	IGUN	42
b.	IPNCH	42
c.	IEXT	42
d.	ISAM	43
e.	IMP	43
f.	KER	43
4.	Flight and Weapon Parameter List	43
a.	X, Y, Z	43
b.	CA, HDG, RA	43

c.	VEL	43
d.	XDOT, YDOT, ZDOT	44
e.	MNUM	44
f.	MBR	44
g.	T	44
h.	XGUN, YGUN, ZGUN	44
i.	XSAM, YSAM, ZSAM	44
j.	GR	44
5.	Simulation Parameter List	45
a.	TMD	45
b.	ACLIFT	45
c.	CLMAX	45
d.	WL	45
e.	TMAX	45
f.	CDO	45
g.	CDK	45
h.	VMAX1, VMAX2	46
i.	APPMAX	46
j.	HTMIN, HTMAX	46
k.	SPDMIN	46
l.	GMAX	46
m.	POPIN	46
6.	Coordinate List	46
a.	X1, Y1, Z1	47
b.	V1	47
c.	LTR, LTR2	47

7.	Error Message Pointer	47
a.	MKERX	47
b.	MKERY	47
B.	EXTERNAL DATA SETS	48
1.	Weapon Locations	48
2.	Map File	48
3.	Milestone File	48
a.	Elements 1-4	48
b.	Element 5	48
4.	Output Files	49
5.	Rule Files	49
C.	MODULES	49
1.	MAIN	49
2.	VALSET	55
3.	ERRCHK	58
4.	PTHPLT	62
5.	SPOT	64
6.	SCENE	64
7.	WIN	68
8.	XYZIN	68
9.	GUNLOC	70
10.	BEGIN	72
11.	GUNCHK	72
12.	PRESET	75
13.	ELFIN	77
14.	AIMPT	79

15.	CONCHG	81
16.	ERRMK	81
17.	SAMCHK	83
VII.	MODIFICATION INSTRUCTIONS	87
A.	GENERAL	87
1.	Screen and Virtual Window Definition	87
2.	Move and Draw	87
3.	Print to the Graphics Screen	88
4.	Cursor Input	88
B.	THE KEYBOARD VERSION (KBPIP)	90
1.	MAIN	90
2.	ERRMK	90
3.	PTHPLT	91
4.	GUNLOC, WIN	91
5.	SPOT	91
6.	SCENE	91
7.	XYZIN	91
8.	BEGIN	92
9.	AIMPT	92
10.	ELFIN	92
11.	CONCHG	92
C.	THE IBM GRAPHICS VERSION (IBMPIP)	92
1.	PTHPLT, GUNLOC, SCENE, WIN, ELFIN, AIMPT, SPOT . .	92
2.	BEGIN	92
3.	CONCHG	92
4.	ERRMK	92

VIII. MAP GENERATION	94
IX. PROBLEMS AND IMPROVEMENTS	96
APPENDIX A PIP USER'S MANUAL	100
APPENDIX B GRPIP PROGRAM LISTING	127
APPENDIX C KBPIP MODIFIED MODULES	148
APPENDIX D IBMPIP MODIFIED MODULES	159
APPENDIX E SCENE PROGRAM LISTING	168
LIST OF REFERENCES	169
BIBLIOGRAPHY	170
INITIAL DISTRIBUTION LIST	171

LIST OF TABLES

1. MODULE GRAPHICS REQUIREMENTS	89
-------------------------------------------	----

LIST OF FIGURES

1. Components of Average Velocity--Horizontal Flight	28
2. Aircraft and Reference Coordinate Systems	30
3. TEKTRONIX 4662 Plot	36
4. TEKTRONIX 4012 Plot	37
5. IBM 3277-based Graphics Plot	38
6. MAIN Flow Chart	52
7. VALSET Flow Chart	57
8. ERRCHK Flow Chart	60
9. PTHPLT Flow Chart	63
10. SPOT Flow Chart	65
11. SCENE Flow Chart	67
12. WIN and XYZIN Flow Charts	69
13. GUNLOC Flow Chart	71
14. BEGIN Flow Chart	73
15. GUNCHK Flow Chart	74
16. PRESET Flow Chart	76
17. ELFIN Flow Chart	78
18. AIMPT Flow Chart	80
19. CONCHG Flow Chart	82
20. ERRMK Flow Chart	84
21. SAMCHK Flow Chart	86

I. INTRODUCTION

Simulating an aircraft flight path requires a great deal of information. This information usually includes position and velocity as a function of time. More detailed descriptions may require heading, climb, and roll angles, as well as velocity components in one or more coordinate systems. Additionally, the simulation will have constraints on the aircraft's performance, such as G load and power available. Finally, the simulation will have certain aspects of aircraft utilization such as altitude or approach limits, which it tests. These might relate to a bombing mission, transcontinental passenger service, or an encounter with an anti-aircraft system. Although real-time flight path simulation is necessary in the final stages of a system's development, the use of simple, pre-defined flight paths in the early design phases is still a helpful tool.

Generating the information for these flight paths is an excellent application of the capabilities of the digital computer. Given a few broad parameters, the computer can generate and properly format the other information the particular simulation requires. The alternative is the time-consuming and error-prone process of having the simulation user determine and properly enter all of the data needed for the simulation.

To further aid the user, a graphic capability is desirable. It is much simpler for a user to decide upon a flight path if he can see where he is flying. In addition, cursor-mapped input reduces the chances of incorrectly typing in parameters.

There are two simulation programs at the Naval Postgraduate School which require pre-defined aircraft flight paths as input data. They are the Air Force Armament Laboratory's Anti-Aircraft Artillery (AAA) Simulation (called P001) and the Vought Corporation's surface-to-air missile (SAM) engagement simulation (called MICE II). Both programs are used by the military aircraft industry, being specifically called for in Reference 1. Both programs are used in the Naval Postgraduate School course in aircraft survivability (AE 3251) as tools for gaining an understanding of the factors affecting aircraft survivability in a hostile AAA and SAM environment. The students are required to fly a bombing mission against a target defended by seven AAA weapons, one which has a fixed location, and one SAM site. They are formed into teams of two, and each team plots a flight path and selects defensive positions. One team then flies its route against an opposing team's weapons, and vice versa.

At the present time, P001 and MICE II are implemented for batch processing on the school's IBM 3033 computer, having been converted from the IBM 360 system in 1980. The input to both programs requires a significant amount of formatted data, the entry of which is time-consuming, tedious, and error-prone. To alleviate part of this problem, a pre-processing program called PIP (P001 Input Processor) was written in 1978. This program required that the operator enter only the milestone X, Y, and Z coordinates, an initial speed, and control indicators for the type of output desired. It too was written for the IBM 360 system batch mode. It was converted to the new 3033 system in 1981, but was not re-written.

With the improved time-sharing capabilities of the 3033 system, it was desirable to make PIP interactive. A significant problem of the

batch system was the requirement to submit a complete flight path to PIP for evaluation with respect to aircraft performance and mission rules. PIP evaluated the milestones and indicated which ones were outside the allowable limits of the parameters. The student then had to change the milestones that were causing the problems and hope that the corrections did not affect the points that were not changed. Since the run often generated punched cards for P001 and MICE, a great deal of waste occurred. Another problem was that the user had to plot the points on a map, copy them to a piece of paper, and then type or punch them in using the proper FORTRAN format. This was extremely tedious and sensitive to typing errors.

The major goal of this thesis effort was to alleviate these problems through the use of an interactive, graphics capability. The availability of TEKTRONIX and IBM graphics terminals and software, and the IBM 3033 CMS made this possible. Using a graphic terminal, the user is presented with a map of an attack scenario. A cursor is used to input the hostile AAA and SAM locations. The cursor is also used to input the aircraft milestone position and velocity. As each milestone is entered, the flight parameters are computed and checked against flight (maximum velocity, G loading, etc.) and simulation (bomb drop distance, pop-up maneuver, etc.) constraints for validity. Milestones which do not meet the constraints are rejected, the user is informed of the error, and a new milestone is requested. At any time up to the final milestone the user may reset the problem to any previous point. The user may also input data from an existing file to re-run a previous trial. When the final milestone is identified, the operator is given the option of

generating P001 and MICE input data files, or no file at all. In addition, the gun, SAM, and flightpath data are saved for later use.

A secondary goal was that of software transportability. The intention is to make the programs available to the survivability community. To support this goal, a modular design was used. Routines which were software or hardware dependent were modularized for ease of replacement. Requirements unique to the user (simulation rules, vulnerability tables, etc.) were also modularized. This modularization allowed the creation of a non-graphic version that would work at any keyboard terminal.

The main body of this thesis will describe the functions of the various modules. This will include a description of parameters passed and common blocks required.

II. APPROACH

A. GOALS

As stated above, the problem was approached with two goals in mind: ease of interaction and transportability. Ease of interaction meant that the system should require minimum interaction from the user while providing rapid feedback to required inputs. Transportability meant that users on other systems could easily convert the program to their systems.

To support the goal of simple interaction, a graphics capability was felt to be vital. This would allow the user to get a much better feel for the data he was creating. In addition, the burden of formatting and copying would be eliminated. Once the data has been input, the program must be prompt in providing feedback as to its validity. A major concern in interactive systems is the response time. For direct data input, a response time of less than one second is desirable. For inputs requiring further computation, two to three seconds is an adequate time. For complicated actions, such as drawing a map, responses on the order of minutes is appropriate. These times are primarily based on the human willingness to accept delay.

To meet the requirements of transportability, several changes to PIP were necessary. To support different hardware and software configurations, any section which relied on specific system characteristics had to be isolated for easy replacement. This applied particularly to the graphics sections, which presently require either the TEKTRONIX 4010 family of terminals and PLOT-10 software, or IBM graphic terminals and

GRAF77 software. By isolating these modules in FORTRAN subroutines, the system will operate with other graphic systems if the appropriate software is changed. By eliminating the graphic commands, the system can be converted to run on any keyboard terminal.

Another area where changes are needed is in the sections dealing with simulation rules. The rules presently included are those which support the Naval Postgraduate School aircraft survivability course. The flight parameters are computed from the user input of position and velocity. Other users may desire to incorporate more or less sophisticated rules and parameter computation to better meet their needs. To accomplish this, two subroutines and the BLOCK DATA must be changed.

B. METHODOLOGY

In planning the program, the basic logic sequence was first outlined. This allowed the preliminary definition of the various modules. A module was created any time a function either occurred more than once, or required implementation unique to a particular system. For example, the ability to read position coordinates was needed several times, so the module XYZIN was written. Similarly, providing information to the user to assist in aligning his flight path properly for a bomb run occurs in only one location, but the manner in which the data is displayed depends on the type of terminal being used.

Once the modules were defined, it was possible to determine which data would be shared between them. When several modules required the same data, or a parameter list would be extremely long, a global data set was created to pass the information.

Finally, the system was implemented with a top-down approach. Stubs were used by higher level modules until lower level modules could be written and tested. This approach reduced the code that had to be checked after a particular test when an error was discovered.

III. FUNCTIONAL DESCRIPTION

A. DATA SETS

1. Internal Sets

a. Screen Window Defined

This data set must contain the coordinates of the various screen windows which the program will use.

b. Target Definition

This data set contains the virtual screen coordinates of the target.

c. Options List

This data set contains the variables necessary to transmit user options throughout the program.

d. Flight Parameter List

This data set contains the variables which define the flight path.

e. Simulation Parameter List

This data set contains the variables which are the constraints on aircraft performance that the user wishes to incorporate.

f. Coordinate List

This data set contains the variables associated with the user's entry of milestones.

g. Error Message Pointer

This data set contains the information necessary to locate the error messages.

2. External Sets

a. Weapon Locations

This data set contains the location of the hostile weapon sites.

b. Map File

This data set contains the X/Y coordinate pairs and instruction markers for drawing the attack scenario map.

c. Milestone File

This data set contains the X, Y, and Z coordinates, velocity, and option commands for each milestone of the flight path.

d. Output File

There is a separate file for each simulation for which the program provides data. In addition, a scratch file is maintained in case of system failures.

e. Rule File

This file saves the simulation constraints to allow the user to rapidly change the entire set of constraints.

B. MODULES

1. MAIN

This module is the sequence control routine and is invoked on program load. The module calls BEGIN to request the user's options. It then calls SCENE to draw the scenario. Based on user response during initialization, the module then establishes the weapon locations by reading a disk file, using a default set, or having the user enter them at the terminal. If the terminal input option is selected, the old weapon site file is erased and the new values are placed in it. MAIN

then accepts data for the aircraft flight path milestones. The user decides during initialization whether the data will come from a disk file or the terminal and whether or not errors will cause the inputs to be rejected. Once the data for each milestone is read, the flight parameters are computed in the VALSET module and checked for errors in the ERRCHK module. Errors are identified to the user and the data is rejected if the user so indicated. If no errors occur, or if they are being ignored, the module calls the PTHPLT module to draw the current leg. When using the terminal to enter the milestone data, the user may, at any point before the final milestone, reset the flight path to a previous point and begin from there to enter new data. If the input disk file has a continuation command, the program will shift from the disk input mode to the terminal input mode. Once the final entry is indicated, MAIN calls the ELFIN module to obtain the user's output options and then calls the PRESET module to implement them.

2. VALSET

VALSET is the module responsible for computation of those parameters required by the main simulation programs but not provided by the user. It contains the equations necessary for the computation of those parameters. The module can also indicate an error when the user's data causes a condition which will be undefined, such as inputs which cause a division by zero.

3. ERRCHK

This module contains the rules that the user wishes the flight path to obey. These include constraints on flight performance (thrust, acceleration, etc.) and rules particular to the mission (approach rules,

bomb release limits, etc.). A violation returns an error code identifying the rule that was violated.

4. PTHPLT

This is the module responsible for notifying the user that the data entered met all the constraints and has been accepted. It also saves the user's input in a scratch file to protect against system crashes.

5. SPOT

This module is the actual interface between the user and the computer for entering position data. It obtains two positions (X and Y) and two command inputs from the user and transmits them to the program. The calling routine is responsible for properly interpreting the X and Y values.

6. SCENE

SCENE is responsible for drawing the graphic displays needed by the program. Data for the map is read from the map file. The data is either a move or stop command or an X/Y pair indicating where the beam is to be positioned. The module also has a role in the reset sequence. When a reset procedure is executed by MAIN, it passes the information to SCENE. SCENE then prompts the user to indicate which milestone is the last one he wishes to retain. This value is returned to MAIN.

7. WIN

This module is responsible for defining a window on the screen. The module is provided with the screen coordinates desired and the virtual range that the window represents. The lower left corner is always given a virtual value of (0,0). Under control of the calling

routine, the module will flash the window to act as a prompt to the user to look at that window.

8. XYZIN

This module is responsible for obtaining the data necessary to identify a single location. The module defines the windows by calling WIN and actually obtains the data by calling SPOT.

9. GUNLOC

This module is used to display special points such as the weapon locations and the target. The location is marked with a "+". It then draws a circle around the location with a radius specified by the calling routine.

10. BEGIN

BEGIN is the module responsible for initializing the program. The user's options for data input are requested and passed to MAIN through the Option Data Set. The module also executes the calls that initialize the graphics routines and computes the coordinates of the various windows to be used.

11. GUNCHK

This module is responsible for checking the gun emplacement rules. A violation results in the appropriate error code being returned.

12. PRESET

This module is responsible for formatting the data so that it is compatible with the data file expected by the simulation program which is to use the flight path. It is also responsible for conversion from Metric to English units where necessary.

13. ELFIN

This module obtains the user's output options and closes access to the graphics routines.

14. AIMPT

This module provides a display to the user to allow proper alignment for the bombing runs.

15. CONCHG

This module allows the user to change the value of the simulation limits. It also saves these changes and allows the user to read a complete set of limits from a disk file.

16. ERRMK

This module sends error messages and prompts to the operator when graphic routines are being used. The message to be set is identified by a parameter passed by the calling routine.

17. SAMCHK

This module is responsible for checking for violations of the SAM emplacement rules. If errors are detected, it calls ERRMK with the appropriate code and returns a "1" to the calling routine. A zero is returned if no errors occur.

IV. FLIGHT DYNAMICS

When defining a flight path, the user must provide a finite set of milestone coordinates. The leg between each pair of milestones is usually considered to be straight. However, this straight line path causes problems with the computation of some of the flight path parameters. For instance, the heading at milestone I in a simple simulation could easily be computed as:

$$\text{Heading}_{ij} = \text{Arctan} (DY/DX)$$

where DX and DY are the X and Y components of the leg from milestone I-1 to I. If "i" and "j" are redefined as I and I + 1, the equation is still a legitimate approximation of the heading at I. In reality, the aircraft's heading at milestone I is somewhere between these two values. The same is true of the climb angle. These two angles in turn define the X, Y, and Z components of velocity at the milestone.

To account for this ambiguity, the program uses an averaging technique for computing the data required to define the flight path. The technique uses the operator's input of position and velocity over three milestones, except at the initial and final milestone, to compute average values of heading, climb, and velocity components.

First, the previous leg's distance and average velocity (from I-2 to I-1) are saved in DISTL and VAVGL for later use. Next, the current leg (from I-1 to I) is evaluated, with the X, Y, and Z components and distance DIST being computed as follows:

$$DX = X(I) - X(I-1)$$

$$DY = Y(I) - Y(I-1)$$

$$DZ = Z(I) - Z(I-1)$$

$$DIST = (DX^2 + DY^2 + DZ^2)^{1/2}$$

The average velocity VAVG equation is:

$$VAVG = (VEL(I) + VEL(I-1))/2$$

The time for the leg (DT) and the total time (T(I)) is then computed:

$$DT = DIST/VAVG$$

$$T(I) = T(I-1) + DT$$

The average X, Y, and Z components of velocity along the leg (I-1 to I) become:

$$AXD(I) = VAVG*DX/DIST$$

$$AYD(I) = VAVG*DY/DIST$$

$$AZD(I) = VAVG*DZ/DIST$$

At milestone I, the acceleration components can now be computed for milestone I-1 (see Fig. 1):

$$DELT = (T(I) - T(I-2))/2$$

$$XDD = (AXD(I) - (AXD(I-1)))/DELT$$

$$YDD = (AYD(I) - (AYD(I-1)))/DELT$$

$$ZDD = (AZD(I) - (AZD(I-1)))/DELT$$

Three working vectors, TDX, TDY, and TDZ, determine the average X, Y, and Z components that will be used to compute the climb and heading angles. These in turn will determine the components of velocity (XDOT, YDOT, ZDOT) at the previous (I-1) milestone. The equations are:

$$HDAVG = DIST/(DIST + DISTL)$$

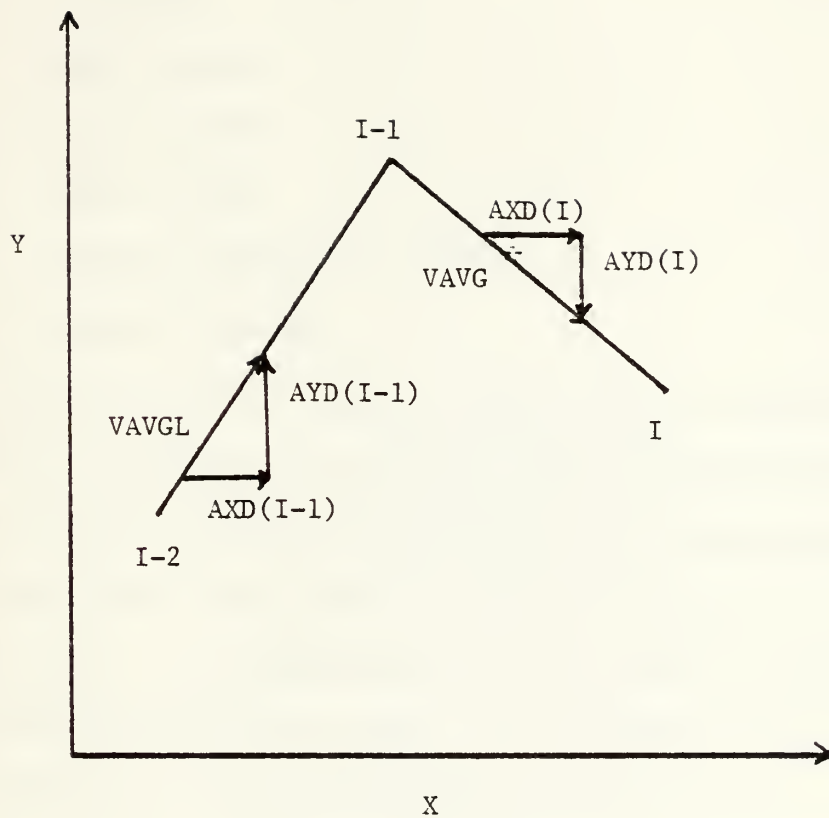


Figure 1. Components of Average Velocity--Horizontal Flight.

$$\begin{aligned}
TDX &= (AXD(I)/VAVG - AXD(I-1)/VAVGL)*HDAVG + AXD(I-1)/VAVGL \\
TDY &= (AYD(I)/VAVG - AYD(I-1)/VAVGL)*HDAVG + AYD(I-1)/VAVGL \\
TDZ &= (AZD(I)/VAVG - AZD(I-1)/VAVGL)*HDAVG + AZD(I-1)/VAVGL \\
UNIT &= (TDX^2 + TDY^2 + TDZ^2)^{1/2} \\
TDX &= TDX/UNIT \\
TDY &= TDY/UNIT \\
TDZ &= TDZ/UNIT \\
XDOT(I-1) &= VEL(I-1)*TDX \\
YDOT(I-1) &= VEL(I-1)*TDY \\
ZDOT(I-1) &= VEL(I-1)*TDZ \\
CA(I-1) &= \text{Arctan} (TDZ/(TDX^2 + TDY^2)^{1/2}) \\
HDG(I-1) &= \text{Arctan} (TDY/TDX)
\end{aligned}$$

To compute the roll angle RA and the forces acting on the aircraft, it is necessary to transform the acceleration from the group coordinate system into an aircraft centered coordinate system with axes parallel and perpendicular to the flight path. The parallel component in the direction of flight is TMD. PH is perpendicular to the flight path, parallel to the X/Y plane, and out the left wing. PP is parallel to the cross product TMD x PH (see Fig. 2). The transformation equations are:

$$TMD = A11 + A12 + A13$$

$$PH = A21 + A22 + A23$$

$$PP = A31 + A32 + A33$$

where

$$GEE = 9.82, \text{ the gravitational constant}$$

$$HDCOS = \cos (HDG)$$

$$HDSIN = \sin (CA)$$

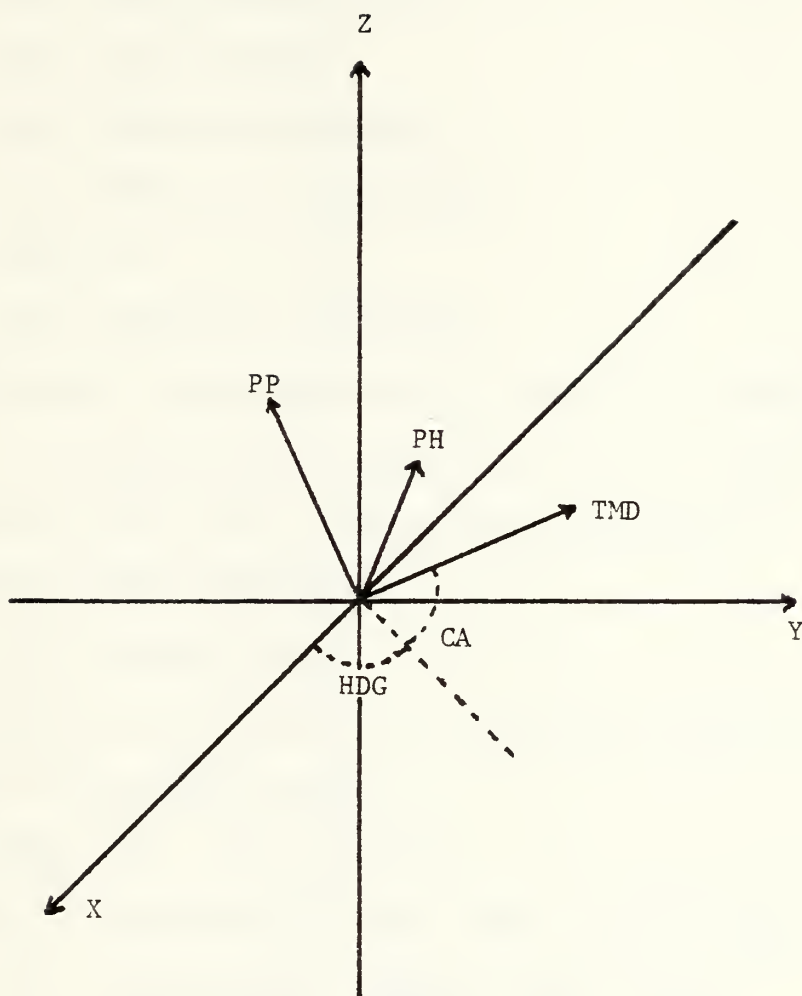


Figure 2. Aircraft and Reference Coordinate Systems.

$$\text{CACOS} = \cos (\text{CA})$$

$$\text{CASIN} = \sin (\text{CA})$$

$$\text{A11} = \text{XDD}(\text{I}-1) * \text{HDCOS} * \text{CACOS}$$

$$\text{A21} = \text{XDD}(\text{I}-1) * \text{HDSIN}$$

$$\text{A31} = \text{XDD}(\text{I}-1) * \text{HDCOS} * \text{CASIN}$$

$$\text{A12} = \text{YDD}(\text{I}-1) * \text{HDSIN} * \text{CACOS}$$

$$\text{A22} = \text{YDD}(\text{I}-1) * \text{HDCOS}$$

$$\text{A32} = \text{YDD}(\text{I}-1) * \text{HDSIN} * \text{CASIN}$$

$$\text{A13} = (\text{ZDD}(\text{I}-1) + \text{GEE}) * \text{CASIN}$$

$$\text{A23} = 0$$

$$\text{A33} = (\text{ZDD}(\text{I}-1) + \text{GEE}) * \text{CACOS}$$

The total lift ACLIFT is the vector sum (PH + PP) of the perpendicular forces and it is assumed that the aircraft will roll such that the wings are perpendicular to the lift. The equations are therefore:

$$\text{ACLIFT} = (\text{PH}^2 + \text{PP}^2)^{1/2}$$

$$\text{RA} = -\text{Arctan} (\text{PH}/\text{PP})$$

The vectors PH and PP are not explicitly computed and, with the acceleration in terms of G's, the equations become:

$$\text{TMD} = (\text{A11} + \text{A12} + \text{A13})/\text{GEE}$$

$$\text{ACLIFT} = ((\text{A21} + \text{A22})^2 + (\text{A31} + \text{A32} + \text{A33})^2)^{1/2}/\text{GEE}$$

$$\text{RA} = -\text{Arctan} (\text{A21} + \text{A22})/(\text{A31} + \text{A32} + \text{A33}))$$

TMD, the net acceleration parallel to the flight path, is the sum of the aircraft's thrust and drag divided by aircraft weight:

$$\text{F}/\text{W} = \text{ma}/\text{W} = \text{ma}/\text{mg} = \text{a}/\text{g} = \text{TMD}$$

where F = total force, W = total weight, m = aircraft mass, a = acceleration, and g = gravitational acceleration. The drag force can be computed from the lift. The equations are as follows:

$$\begin{aligned} \text{RHO} &= 0.256 * e^{(-0.103 * Z(I) / 1000)} \\ \text{CL} &= 2 * \text{ACLIFT} / (\text{RHO} * \text{VEL}(I)^2 / \text{WL}) \\ \text{DW} &= ((\text{CDO} + \text{CDK} * \text{CL}^2) / \text{CL}) * \text{ACLIFT} \\ \text{TW} &= \text{TMD} + \text{DW} \end{aligned}$$

where RHO = density of air in $(\text{lb} * \text{sec}^2) / (\text{ft}^2 * \text{meter}^2)$. The mixed units are required to use Metric units for velocity and English units for wing loading.

$$\begin{aligned} \text{CL} &= \text{lift coefficient} \\ \text{WL} &= \text{wing loading in } \text{lbs} / \text{ft}^2 \\ \text{DW} &= \text{drag divided by weight} \\ \text{CDO} &= \text{drag coefficient for zero lift} \\ \text{CDK} &= \text{factor relating lift and drag coefficients} \\ \text{TW} &= \text{thrust divided by weight} \end{aligned}$$

Maximum values of ACLIFT, CL, and TW were used as constraints on aircraft performance. TW must be positive or zero. If it is found to be negative, then TMD must be negative. This is a large negative acceleration and implies speed brakes are being used. Consequently, a different drag equation must be used. The program assumes the brakes have a drag coefficient of one and a surface area equal to five percent of the wing surface. The new drag equation is:

$$\text{DW} = (\text{ACLIFT} / \text{CL}) * (0.05 + \text{CDO} + (\text{CDK} * \text{CL}^2)) / 2$$

This drag is used to reevaluate TW and thus determine whether or not the attempted deceleration exceeded the capabilities of the speed brake.

V. GRAPHICS HARDWARE AND SOFTWARE

Two interactive graphic systems are supported by the IBM 3033 computer. The TEKTRONIX PLOT-10 software package supports terminals from the TEKTRONIX 4010 family and is available for implementation on a wide variety of computers, including minicomputers. The IBM 3277-based graphics package is available only on IBM computers used in conjunction with modified IBM 3277 terminals. Both systems use direct view storage tubes for the graphics display and keyboard and cursor for data input.

A direct view storage tube is one in which the picture does not have to be continuously refreshed. The major advantages with this type of display is that the line drawing is faster and the lines that are drawn are continuous. This creation of continuous lines is not available on raster scan (television style) display devices. These displays sweep horizontally from top to bottom and have a finite number of elements, called pixels, that are turned either on or off on each sweep. These discrete elements can lead to the "staircase effect," where the line makes finite jumps from one row or column to another. On direct view storage tubes, this effect is eliminated because the electron beam can be steered to any point on the screen and moved directly to any other point.

The major disadvantage of most storage tube displays is that portions of the screen cannot be erased without erasing the entire screen. In order to remove an undesired display, the entire screen must be cleared and then the portions that are to be retained are redrawn. This can be

very annoying to the operator. It also causes delays that adversely affect the user's willingness to continue with the program. For instance, drawing the scenario displays, when operating at 300 BAUD, takes almost five minutes. Raster scan devices do not have this problem because the picture is continuously redrawn and there is no difficulty changing it.

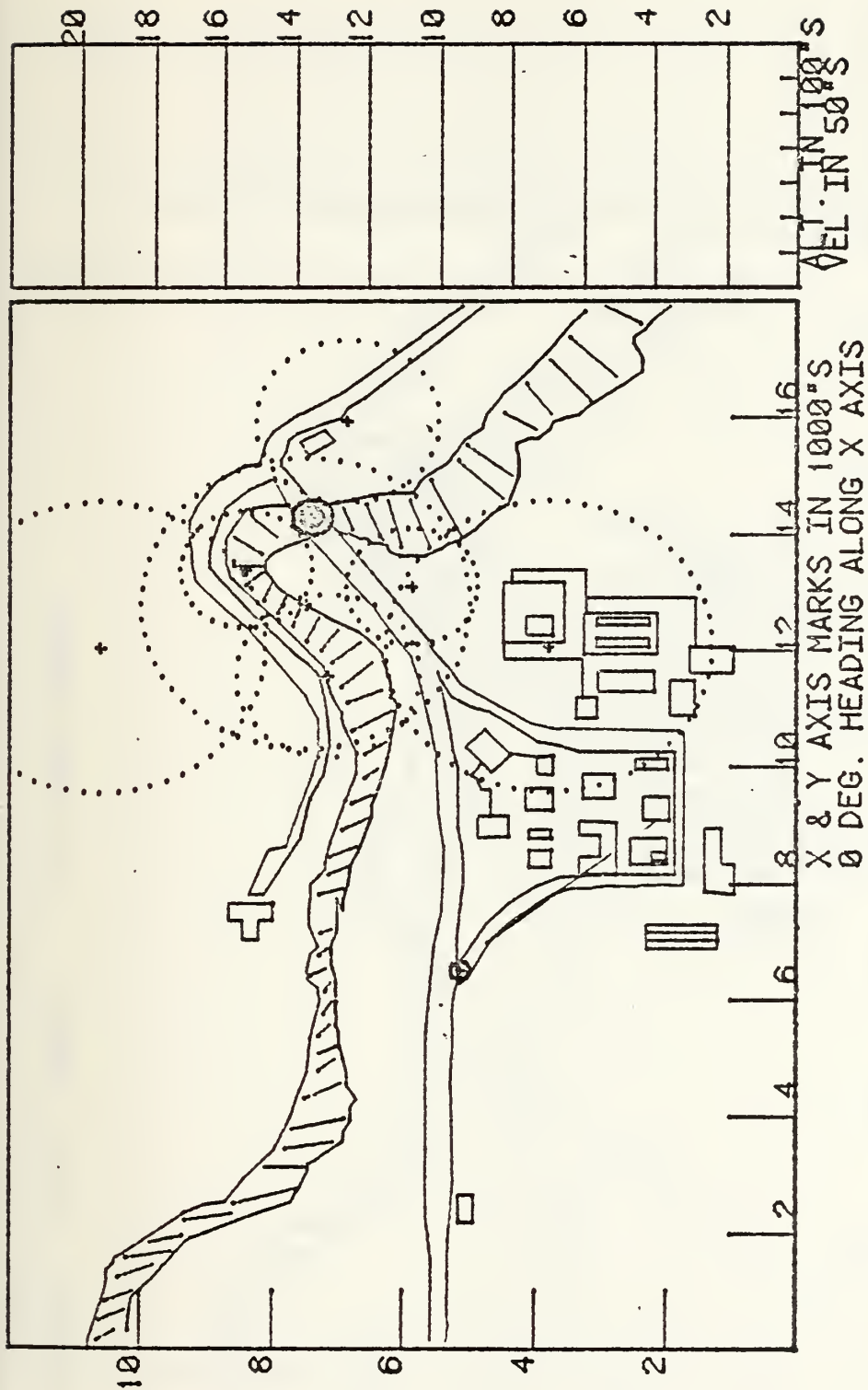
The TEKTRONIX terminal for which this program was written is the Model 4012. This terminal has an 8-1/2 inch by 6-1/2 inch screen and a standard computer keyboard. The PLOT-10 software identifies the screen as a coordinate system 1,023 units wide and 780 units high, with the lower left corner having a value of (0,0). The software also provides for the definition of screen windows and virtual coordinate systems. A screen window is a rectangle on the screen. It is defined by the lower and upper horizontal and vertical coordinates the programmer provides. There is no limit to the number windows that can be defined, but only one exists at a particular time. A similar concept is the virtual window. Once a screen window is activated, the programmer can assign the range of coordinates that this window will represent. The PLOT-10 software also provides for converting from Metric and English units to screen units and vice versa. Combined with a virtual coordinate system, this allows the programmer to provide true scale drawings. For example, the attack map screen window is defined as starting 1/5 inch from the left side of the screen and extending to 6-1/5 inches from the left. This six inch length is then given virtual limits of 0 and 18,000, giving a true scale of 3,000 meters/inch.

Hard copy output from the TEKTRONIX 4012 can be obtained by connecting the terminal to either the TEKTRONIX 4662 interactive plotter or the

TEKTRONIX 4631 hard copy device. The 4662 plotter is capable of being independently controlled with PLOT-10 software, or it can duplicate what is occurring on the 4012 screen. Figure 3 is the output from the plotter operating in parallel with the terminal. The 4012 can also electronically scan the display. This information is then used to control the light intensity that exposes the light sensitive paper in the 4631 copier. Figure 4 is the output obtained from the 4631 hard copy device.

The IBM 3277-based graphics system is a dual screen system. It uses an additional circuit card in an IBM 3277 terminal, which has a raster scan, to drive a TEKTRONIX 622 direct view storage tube display. The software package is very versatile. It provides a wide variety of three-dimensional, geometric, and vector-drawing support and a limited capability for a moving display. Another advantage is the availability of the raster terminal. This greatly facilitates alphanumeric, as opposed to graphic, input and output. The extensive support this package provides is also a handicap. The functions are generally more difficult to use than those provided by PLOT-10. In addition, the software is partially incompatible with FORTRAN terminal input/output. When FORTRAN input/output routines (WRITE or READ) are used, the raster screen must be manually cleared before execution continues. This is a major distraction.

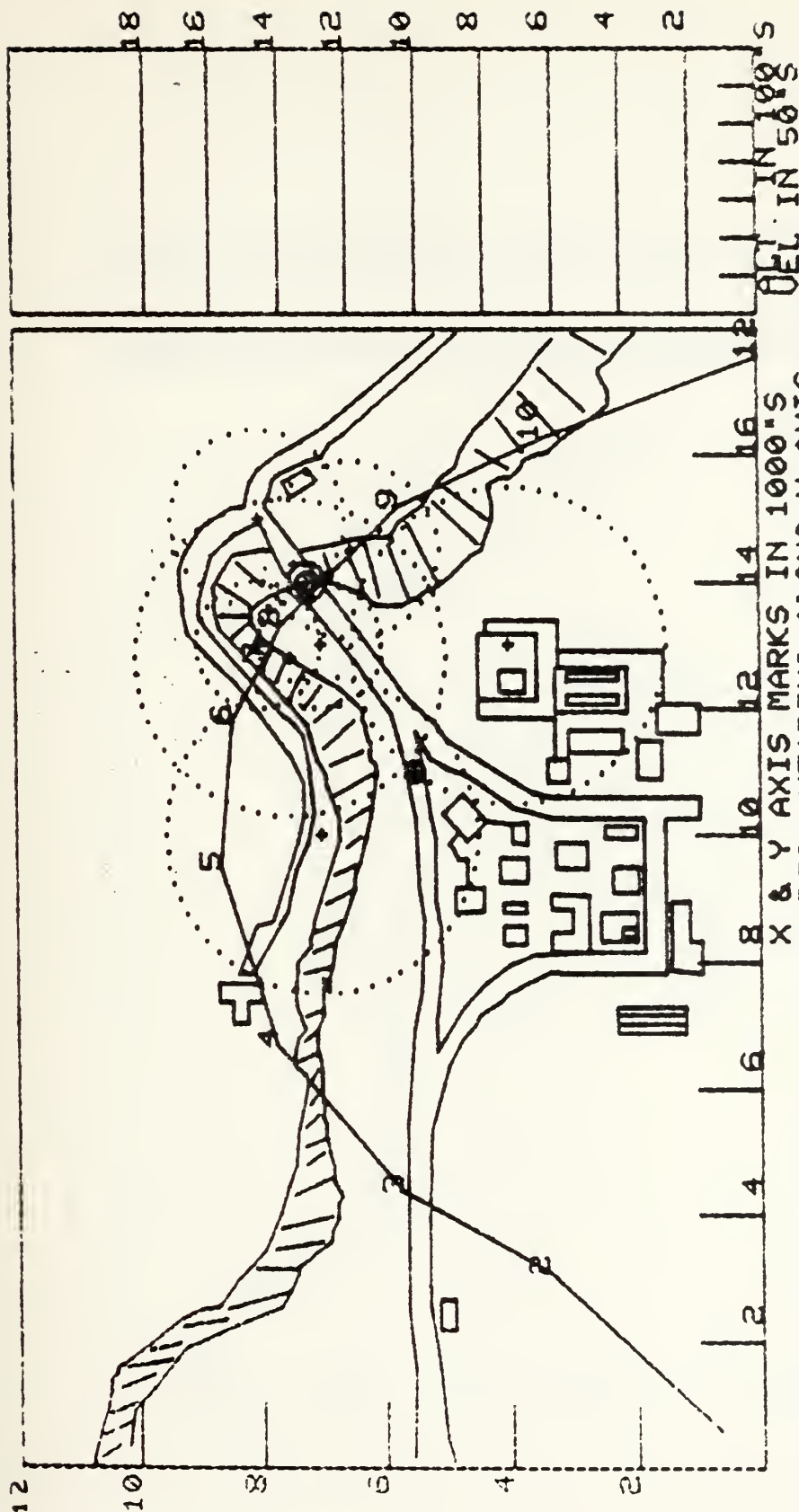
Like the 4012, the TEKTRONIX 622 can electronically scan its display and drive the 4631 hard copy device. Figure 5 is a copy of the display as it appears on the 622 screen. Ultimately, the Naval Postgraduate School's TEKTRONIX 622's will be connected to a VERSATEC electrostatic plotter. This will significantly improve the overall capability of the graphic support at the school.



MAX ACCEL EXCEEDED	BMB DROP LOW	TOO CLOSE TO TGT
MAX G EXCEEDED	BMB DROP HI	ENTER MISSILE LOCATION
APPRCH TOO LOW	TOO FAR FM TGT	ENTER MILESTONES
ALT TOO HIGH	HDG>5 DEG TO TGT	ENTER GUN COORDINATES
STALL	FINAL RUN<2.33 SEC	MAX LIFT EXCEEDED
POP-UP TOO LOW	NO HORIZONTAL MOTION	MAX THRUST EXCEEDED

Figure 3. TEKTRONIX 4662 Plot.

VILLAGE 0-NOT DRAWN, 1- VILLAGE DRAWN



X COORDINATE < 6000
TOO CLOSE TO TGT
ENTER MISSILE LOCATION
ENTER MILESTONES
ENTER GUN COORDINATES
MAX LIFT EXCEEDED
MAX THRUST EXCEEDED
DO YOU WANT TO FIX THE ERROR 0-NO, 1-YES

BMB DROP LOW
BMB DROP HI
TOO FAR FM TGT
HDG>5 DEG TO TGT
FINAL RUN<2.33 SEC
NO HORIZONTAL MOTION POP-UP TOO LOW
ERROR 0-NO, 1-YES

MAX ACCEL EXCEEDED
MAX G EXCEEDED
APPROCH TOO LOW
ALT TOO HIGH
STALL

Figure 4. TEKTRONIX 4012 Plot.

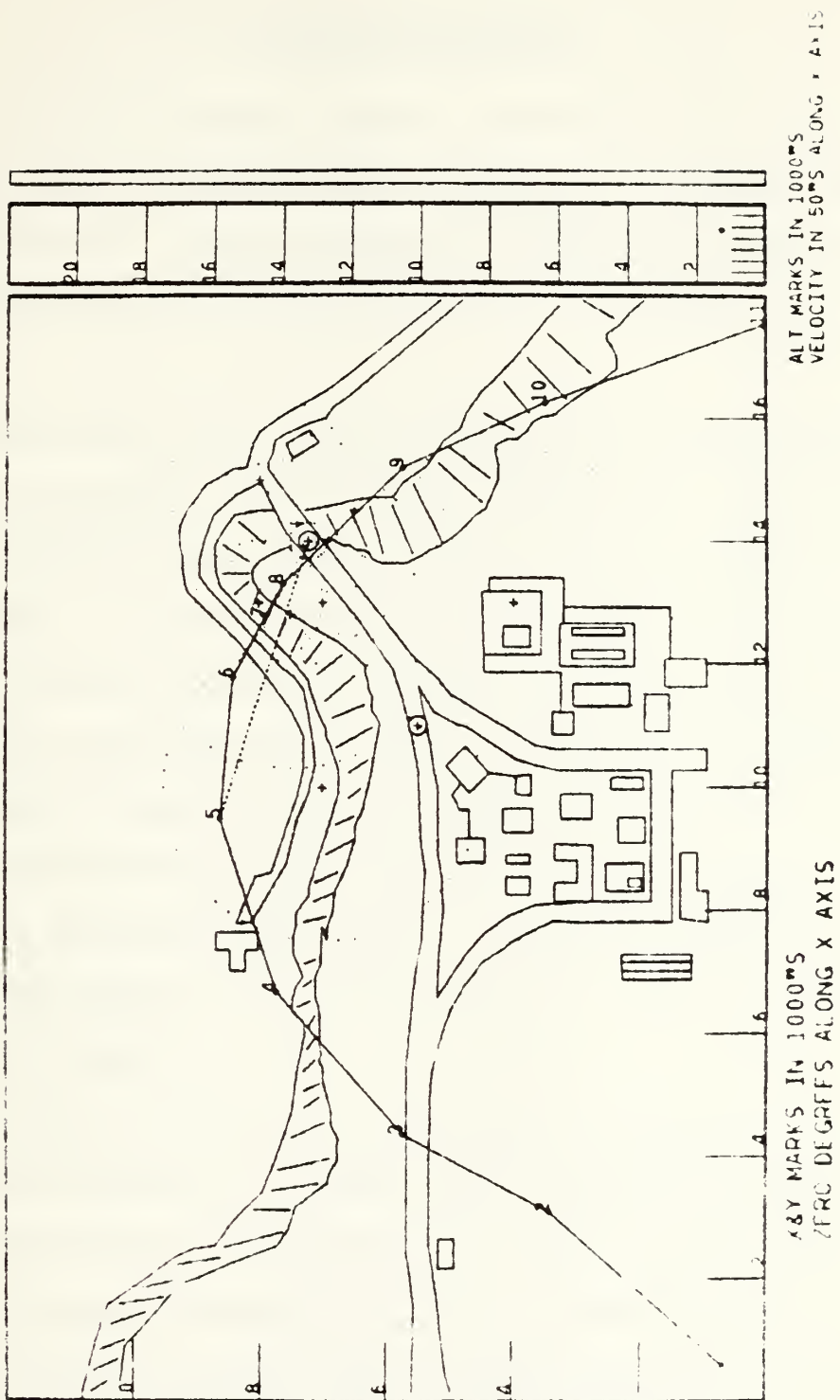


Figure 5. IBM 3277-based Graphics Plot.

VI. DETAILED DESCRIPTION

This section will provide a detailed description of the program as implemented using the PLOT-10 software to support the TEKTRONIX 4010 family of terminals. The first segment will describe the internal and external data required by the program while the second will describe the modules.

For the purposes of this section, a definition of the meaning of "data set" is required. The words "data set" describe a logical group of data. The FORTRAN language uses the term "data set" to identify a hardware device. The term "FORTRAN data set (number)" will be used to define this concept. For example, the MAP file is a data set consisting of points that define the scenario map. The information is contained on a device which has been identified as FORTRAN data set 9.

The description of the internal data will identify the use of each variable in a data set, and the range of values the variable can be assigned. The external data set descriptions will list the contents and format of each element.

The final part of this section will describe the modules. A brief functional description will be given. Those parameters which must be provided to the module or returned by it will then be described. The global data sets which the module uses will be identified, but not the individual elements of the set. Next, the description will list the higher level modules that the module must support. This will be followed by a list of the module's local variables, which will describe the use

and possible values of the variable. The final segment will be an algorithmic and flowchart description of the module. In the algorithmic description comments will be marked by square brackets ([]).

A. INTERNAL DATA SETS

1. Screen Window Defined

The variables for this data set are contained in common block MN.

The variables are:

a. IBAUD

This variable identifies the baud rate of the user's terminal. This is necessary to allow the screen prompts to flash sufficiently long to attract attention, but not too long as to be distracting. The allowable values are: (1) 0--this indicates a slow (110-300 BAUD) terminal interface; and (2) any positive integer--this indicates a fast (1,300-9,600) terminal interface with the computer. The larger the number, the more times the prompt flashes will appear. Since the baud rate determines the length of time between flashes, a larger number of flashes will result in the prompt being visible for a greater period of time.

b. MINY, MAXY

These variable identify the lower and upper vertical limits for all the screen windows. They can be assigned any positive integer value within the limits of a TEKTRONIX screen, subject only to the constraint that MAXY is greater than MINY. For the TEKTRONIX 4012 terminal, the values are: (1) MINY = 777-KIN(4.375), which places the screen windows' lower boundary 4.375 inches from the top of the screen; and (2) MAXY = 777-KIN(0.375), which places the screen windows' upper limit 0.375 inches from the top of the screen, making the windows four inches high.

c. MINX, MAXX

These are the left and right screen limits of the map window. They can be assigned any positive integer value within the limits of a TEKTRONIX screen, subject only to the constraint that MINX is less than MAXX. For TEKTRONIX 4012, the values are: (1) $\text{MINX} = \text{KIN}(0.20)$, which places the left screen boundary of the map one fifth inch from the left edge of the screen; and (2) $\text{MAXX} = \text{KIN}(6.20)$, which places the map's right screen boundary 6.20 inches from the left side of the screen for a window six inches wide.

d. MIN1, MAX1

These variables define the left and right screen borders of the altimeter. They can be any positive integer within the limits of the TEKTRONIX screen, subject to two constraints. MIN1 must be less than MAX1 and the range of MIN1-MAX1 must not overlap that of MINX-MAXX. For this program, the values are computed as follows: (1) $\text{MIN1} = \text{MAXX} + 10$, which places the left screen limit of the altimeter window 10 units to the right of the map window; and (2) $\text{MAX1} = \text{MAXX} + 185$, which places the right border 185 screen units to the right of the map window and makes the altimeter window 175 units wide.

2. Target Definition

The variables for this data set are found in common block TAR. They are:

a. TARGX

This is the X coordinate of the target. It must have a positive value less than 18,000 to appear on the map and is initialized in the Block Data section to a value of 14,000.

b. TARGY

This is the Y coordinate of the target and must be a positive value less than 12,000 to appear on the map. It is initialized in the Block Data section to a value of 7,220.

3. Option List

These variables are found in the common block OPT and are defined as follows.

a. IGUN

This is a variable used to indicate the source of the weapon site data. The values are assigned the following meaning: (1) 0--the weapon site data is to be obtained from the external data set GUN LOC; (2) 1--the weapon site data will be entered by the user at the terminal; and (3) any other value--the weapon site data is the default data contained in common block PAR2.

b. IPNCH

This control variable relays the users choice of the output files to be created by module PRESET. The values mean: (1) 0--no output files are desired; (2) 1--only a P001 file is to be written; (3) 2--only a MICE II file is to be written; and (4) any other value--both P001 and MICE II files are to be written.

c. IEXT

This controls whether or not the P001 extended output option cards are written. The values are: (1) 1--extended output option is requested; and (2) any other value--the option is not requested.

d. ISAM

This is the missile type to be used in the MICE II file. It may be any integer between 1 and 7. The missile type is defined in Reference 2.

e. IMP

This indicates the user's option for the source of the milestone data and has the following definition: (1) 0--the data is to be obtained from the external data set PTS LOC; and (2) any other value--the data is to be entered by the user at the terminal.

f. KER

This indicates the user's option to ignore errors. The possible values are: (1) 0--ignore errors; and (2) any other value--notify the user of violations of the simulation parameters.

4. Flight and Weapon Parameter List

This data set consists of three common blocks. These blocks are PAR, PAR1, and PAR2. The variables in PAR are:

a. X, Y, Z

These are 200 element arrays where X(I), Y(I), Z(I) are the X, Y, and Z coordinates in meters of milestone I.

b. CA, HDG, RA

These are 200 element arrays. CA(I), HDG(I), and RA(I) are the climb, heading, and roll angles in radians of the aircraft at milestone I.

c. VEL

This is a 200 element array with VEL(I) representing the aircraft velocity in meters/second at milestone I.

The variables in PAR1 are:

d. XDOT, YDOT, ZDOT

These are 200 element arrays. XDOT(I), YDOT(I), and ZDOT(I) are the X, Y, and Z components in VEL(I) at milestone I.

e. MNUM

This is a counter that indicates the current milestone.

f. MBR

This is a counter that indicates the milestone at which the aircraft's weapons were released.

The variables in PAR2 are:

g. T

This is a 200 element array. T(1) is zero and T(I) is the elapsed time at milestone I.

h. XGUN, YGUN, ZGUN

These are seven element arrays which contain the X, Y, and Z location, in meters, of the gun emplacements. The default values are initialized in the Block Data section.

i. XSAM, YSAM, ZSAM

These are the X, Y, and Z location, in meters, of the SAM emplacement. The default values are set in the Block Data section.

j. GR

This is a seven element array. GR(I) is the engagement radius of gun I. The values for GR are established in the Block Data section.

5. Simulation Parameter List

The data set is contained in commom blocks PAR3 and PAR4.

PAR3 contains:

a. TMD

This is the net acceleration, in G's, parallel to the aircraft's flight path.

b. ACLIFT

This is the net acceleration, in G's, perpendicular to the flight path of the aircraft.

c. CLMAX

This is the maximum allowable coefficient of lift. It is initialized to 1 in the Block Data section.

d. WL

This is the wing loading of the aircraft in pounds/ft². It is set to 100 in the Block Data section.

e. TMAX

This is the maximum thrust-to-weight ratio, in G's, that the aircraft can provide. The initial value of 0.4 is set in Block Data.

f. CDO

This is the drag coefficient with zero lift. Block Data sets it initially to 0.015.

g. CDK

This is the factor relating the drag and lift coefficients and is set in Block Data to a value of 0.1.

h. VMAX1, VMAX2

These are the maximum allowable aircraft velocities, in meters/second, before and after the bombs have been released and are set equal to 260 and 310, respectively, in Block Data.

The variables contained in PAR4 are:

i. APPMAX

This is the maximum altitude, in meters, that the aircraft can ascend to before executing its pop-up maneuver. Block Data sets this variable to 457.

j. HTMIN, HTMAX

These are the minimum and maximum altitudes, in meters, that the aircraft can position itself. HTMIN is set to 60 and HTMAX to 2,050 in Block Data.

k. SPDMIN

This is the aircraft stall speed, in meters/sec. It is set to an initial value of 90 in the Block Data section.

l. GMAX

This is the maximum G force that the aircraft can sustain and is set to 6 in Block Data.

m. POPMIN

This is the minimum distance to the target, in meters, before the aircraft can begin the pop-up maneuver. Block Data sets it at 6,000.

6. Coordinate List

The variables in this data set are contained in common block LOC. The variables are:

a. X1, Y1, Z1

These are the X, Y, and Z values entered by the user at a milestone.

b. V1

This is the velocity selected by the user.

c. LTR, LTR2

These are operator commands. They have the following meaning:

(1) 65--this indicates that the user wants assistance in aligning his flight path for a bombing run; (2) 66--this means the user wants to release the bombs at this milestone; (3) 85--this means that the user wants to reset the flight path to an earlier point; (4) 86--this means that the user is finished with milestone entry; and (5) any other value--no effect.

7. Error Message Pointer

The variables in this data set are found in common block ERR.

They are:

a. MKERX

This is a three element array that indicates which column contains the error message. For the TEKTRONIX 4012, these columns are set in Block Data to screen coordinates 640, 340, and 40.

b. MKERY

This is a 20 element array that contains the vertical screen coordinates of the error messages. The array is initialized in Block Data.

B. EXTERNAL DATA SETS

1. Weapon Locations

This data set is maintained on FORTRAN data set 10. The data set is written using (3F10.2) format. The three elements are the X, Y, and Z coordinates of the weapon sites. The set contains seven weapon sites. These are the coordinates of gun sites 1-6 and the SAM site.

2. Map File

This data set is found on FORTRAN data set 9. The format is (2F10.2). The information is organized as follows:

1. The first element is the coordinate set that marks the first point to be drawn on the map.

2. Subsequent data is interpreted as follows: (1) positive values--draw a line to the point that was just read; (2) negative value greater than -1--ignore this set, read the next set and move the cursor there without drawing; and (3) negative value less than -1--end of data.

3. Milestone File

This data set is found on FORTRAN data set 11. The format is (4F10.0,13). Each line is organized as follows:

a. Elements 1-4

These are the X, Y, Z, and velocity values for the milestone in (F10.0) format.

b. Element 5

This is the command and is interpreted as follows: (1) 83--last milestone; (2) 66--release the bomb on this milestone; and (3) 1--after this milestone, stop reading milestones from the external data set and begin accepting them from the user.

4. Output Files

These files contain the data in the formats specified in References 3 and 4. The P001 file is written to FORTRAN data set 18 and the MICE II file is written to FORTRAN data set 17.

5. Rule File

This file is found on FORTRAN data set 16. The format is (6F12.4). The first set of six contains the values of CLMAX, WL, SPDMIN, GMAX, HTMIN, and HTMAX. The second set of six contains POPMIN, APPMAX, TMAX, CDO, CDK, and VMAX1. The final set of three contains the values for VMAX2, TARGX, and TARGY.

C. MODULES

1. MAIN

- a. Function: MAIN controls the execution of the entire program.
- b. Parameters Required: None.
- c. Common Blocks: MN, PAR, PAR1, PAR2, OPT, LOC.
- d. Called By: None.
- e. Local Variables:
 - (1) I--used as a loop counter.
 - (2) FTFAC--constant for converting meters to feet.
 - (3) DGFAC--constant for converting radians to degrees.
 - (4) ICT--counter used to indicate the milestone selected

during resets.

(5) LAST--end of file marker for flight path file. A 1 indicates the flight path will be continued, 83 marks the final milestone.

(6) DX, DY, DZ--X, Y, and Z components of the final leg of the flight path, used to compute final values of HDG, CA, XDOT, YDOT, and ZDOT.

(7) BLANK--constant set to 0 for output to flight path file.

f. Algorithm

```
initialize MBR, MNUM, and T(1)
call BEGIN [initialize]
call SCENE [draw map]
rewind gun disk file
if (terminal input of gun sites) then [IGUN = 1]
prompt user to input gun coordinates
do I = 1 to 6
  do until (IERR = 0)
    IERR = 0
    call XYZIN [get coordinates]
    if (Z1 >= 1,000) then STOP
    if (Z1 >= 5) then call GUNCHK [check for valid location]
    return IERR error code]
    if (IERR <> 0) then call ERRMK [indicate error]
  end do
  write X1, Y1, and Z1 to disk file GUNLOC
end do
write missile prompt
do until (IERR = 0)
  call XYZIN [get SAM coordinates]
  if (Z1 >= 1,000) then stop
  call SAMCHK [check for correct placement]
end do
write SAM coordinates to disk file
else if (disk input requested) then do [IGUN = 0]
  do I = 1 to 6
    read XGUN(I), YGUN(I), and ZGUN(I) from disk file GUN LOC
  end do
  read XSAM, YSAM, and ZSAM from disk file GUN LOC
end if [terminal input selected]
do until (command = stop)
  do I = 1 to 7
    call GUNLOC [draw gun site]
  end do
  call GUNLOC [draw SAM site]
  set constants
  prompt user to enter milestones
  rewind milestone disk file PTS LOC
  do until (command = stop or ((command = reset) and (terminal
input requested)))
    increment MNUM
  do until (IERR*KER = 0 and IERR <> 12)
```


FIX

```
if (disk input requested) then [IMP = 0]
  do until (X > 0.1)
    read X, Y, Z, VEL, and LTR [LTR is command
    variable]
  end do
  LTR2 = LTR
  if (command = shift to terminal input) IMP = 1
  [LTR = 1]
else call XYZIN [get X, Y, X, VEL, LTR, and LTR2]
end if
if (MNUM = 1 and command = stop) then STOP [LTR = 83 OR
LTR2 = 83]
if (MNUM > 1) then
  IERR = 0
  call VALSET [compute flight parameters]
  if (command = bomb release) then MBR = MNUM [LTR = 66]
  if (command <> restart and command <> stop) then
    [LTR <> 82 or 83]
    if (IERR = 0) call ERRCHK [return error code IERR]
    if (IERR*KER <> 0 or IERR = 12) then
      call ERRMK
      if (disk milestone input) then ask user if
      error is to be ignored
      if (error not ignored) then go to FIX
    end if
  end if
end if [MNUM > 1]
end do [until IERR*KER = 0 and IERR <> 12]
call PTHPLT [mark milestone]
if (command = aim) call AIMPT [LTR = 65]
end do [until command = stop or command = restart]
if (command = restart) then [LTR = 82]
  call SCENE [redraw map and get restart milestone]
  rewind scratch file TEMP DATA
  if (MBR > ICT) then MBR = 0
  if (ICT = 1) then
    MNUM = 0
  else
    do MNUM = 2 to ICT
      call PTHPLT [plot milestones]
    end do
    decrement NMUM
  end if [ICT = 1]
end if [command = restart]
end do [until command = stop]
call PTHPLT [plot final milestone]
LAST = IMP
call ELFIN [get output options]
compute final milestone flight parameters
if (new flight path or new gun locations created) then
  [IMP = 0, IGUN > 0]
  rewind flight path file PTS LOC
```


MAIN

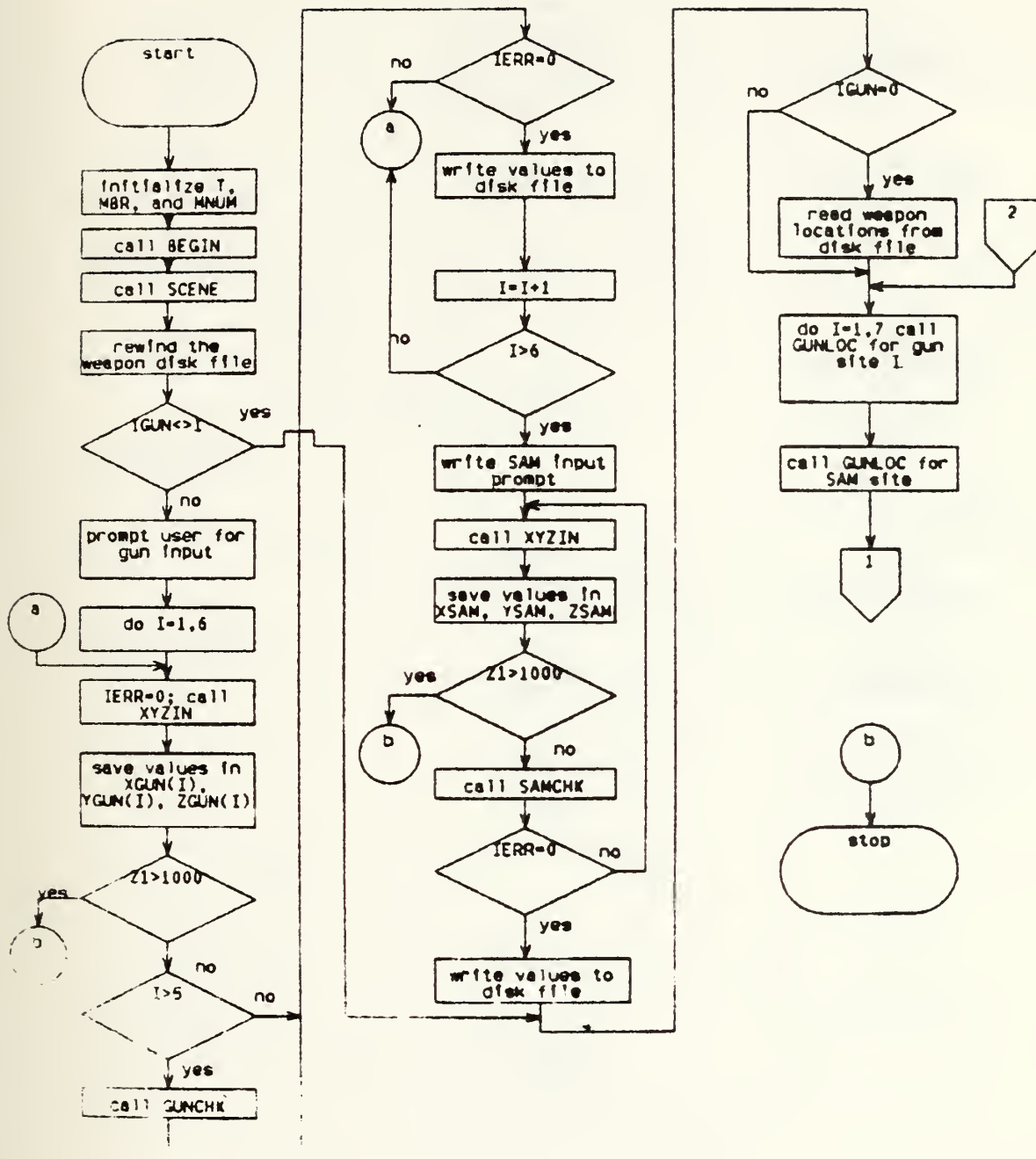


Figure 6. MAIN Flow Chart.

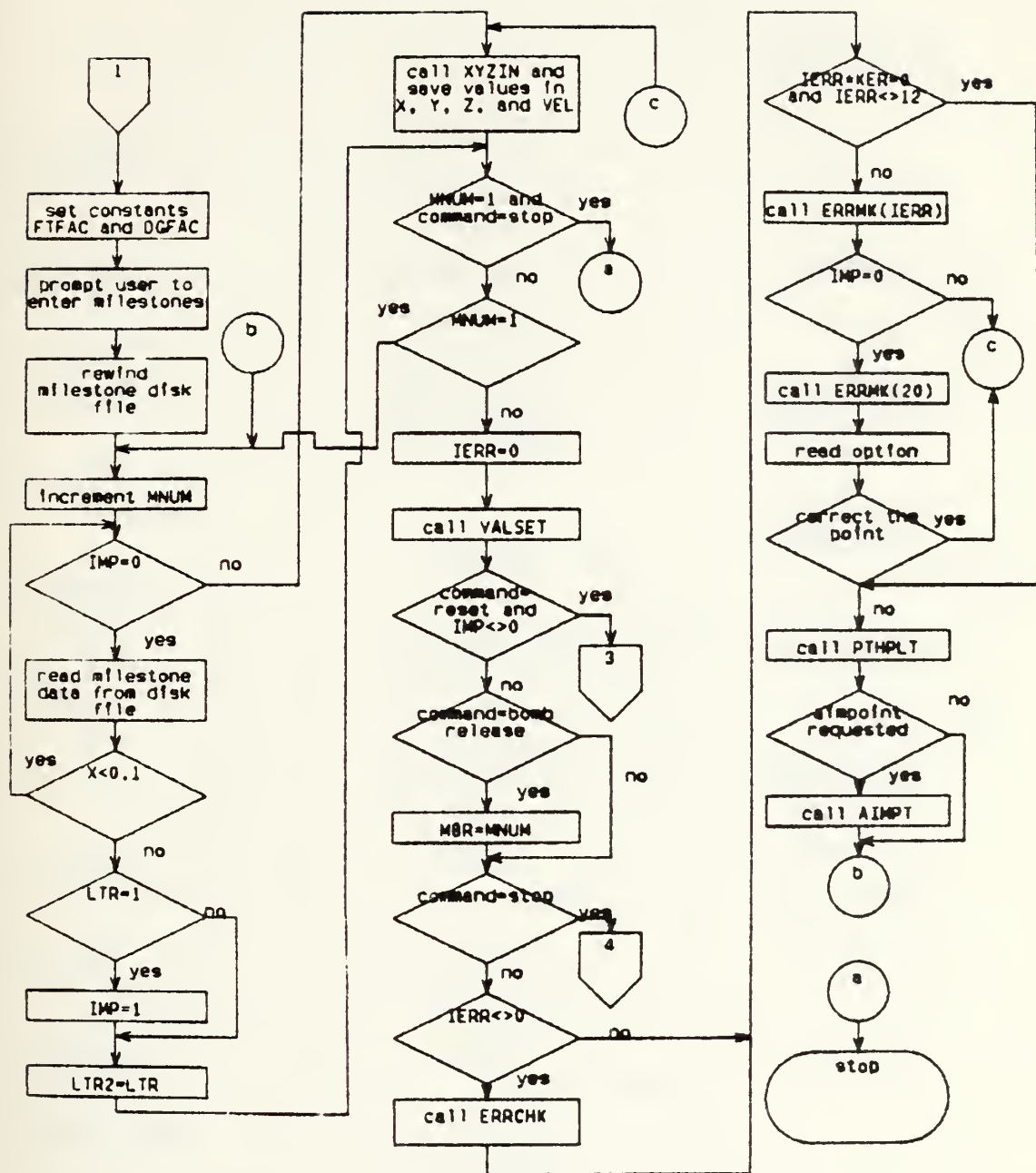


Figure 6 continued.

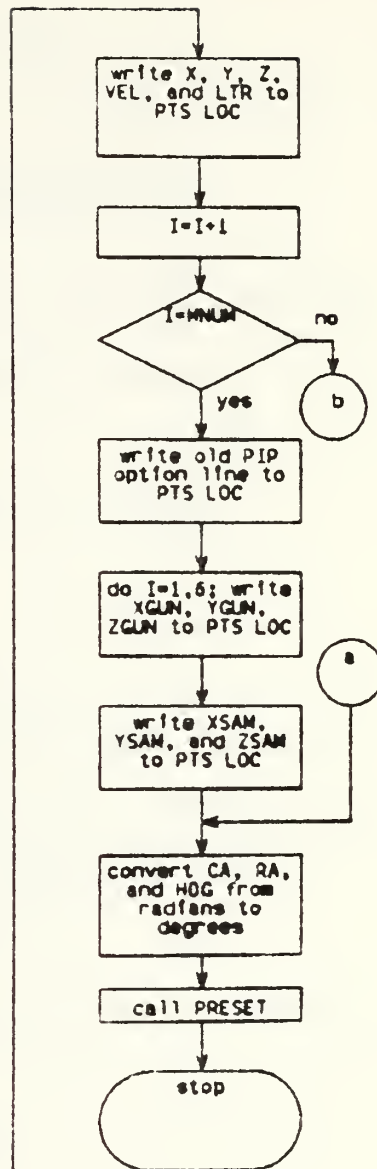
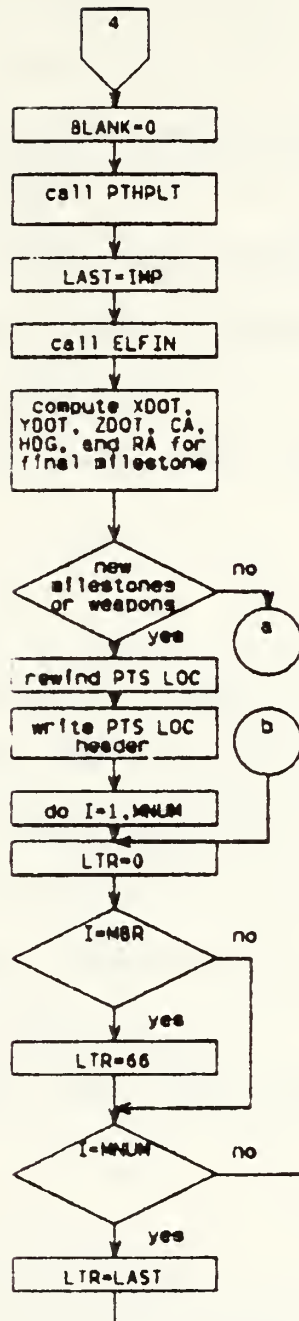
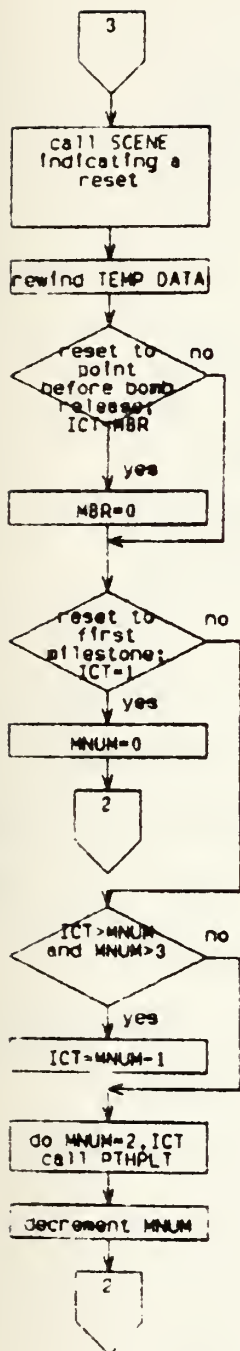


Figure 6 continued.


```

IF (IGUN > 1) then IGUN = 1
write header on PTS LOC
do I = 1 to MNUM
    LTR = 0
    if (MBR = I) LTR = 66
    else if (I = MNUM) then LTR = LAST
    end if
    write to disk file PTS LOC X, Y, Z, VEL, and LTR
end do
write end of milestone marker to file PTS LOC
write old PIP option line to file PTS LOC
if (IGUN <> 0) then
    do I = 1 to 6
        write XGUN, YGUN, and ZGUN to disk file PTS LOC
    end do
    write XSAM, YSAM, and ZSAM to file PTS LOC
end if
end if [IMP = 1 or IGUN > 0]
do I = 1 to MNUM
    convert CA, RA, and HDG from radians to degrees
end do
call PRESET
end MAIN

```

2. VALSET

a. Function: This subroutine computes the flight parameters for each milestone. XDOT, YDOT, ZDOT, CA, HDG, and RA are computed for output to P001 and MICE II. TMD and ACLIFT are computed for use by ERRCHK. See Section IV for the equations of motion.

b. Parameters Required: IERR--set to 12 if there is no change in the horizontal (X, Y) position.

c. Common Blocks: PAR, PAR1, PAR2, PAR3.

d. Called By: MAIN.

e. Local Variables:

(1) GEE--gravitational constant 9.82 meters/sec.

(2) DX, DY, DZ--X, Y, and Z components of the flight-path leg being evaluated.

(3) VAVGL--average velocity of the previous leg.

(4) DISTL--length of previous leg.

- (5) DIST--length of the current leg.
- (6) VAVG--average velocity of the current leg.
- (7) DT--time along the current leg.
- (8) AXD, AYD, AZD--X, Y, and Z components of current average velocity.
- (9) DELT--time along current and previous legs.
- (10) XDD, YDD, ZDD--X, Y, and Z components of acceleration.
- (11) HDAVG--weighting factor for computing TDX, TDY, and TDZ.
- (12) TDX, TDY, TDZ--weighted X, Y, and Z components of velocity.
- (13) UNIT--conversion factor to make (TDX + TDY + TDZ) a unit vector.
- (14) CASIN, CACOS--sine and cosine of the climb angle CA.
- (15) HDGSIN, HDGCOS-- sine and cosine of the heading HDG.
- (16) A11, A12, A13--X, Y, and Z components of acceleration parallel to the flight path.
- (17) A21, A22, A23--components of acceleration perpendicular to the flight path and parallel to the X/Y plane due to X, Y, and Z components of acceleration.
- (18) A31, A32, A33--components of acceleration perpendicular to the flight path not parallel to the X/Y plane due to X, Y, and Z components of acceleration.

f. Algorithm

```

compute DX, DY, and DZ
if (no horizontal motion) then
    IERR = 12
    return
else
    save old values of VAVG and DIST in VAVGL and DISTL
    compute DIST
    if (VEL > VMAX1 and bomb not released) then VEL = VMAX1

```


VALSET

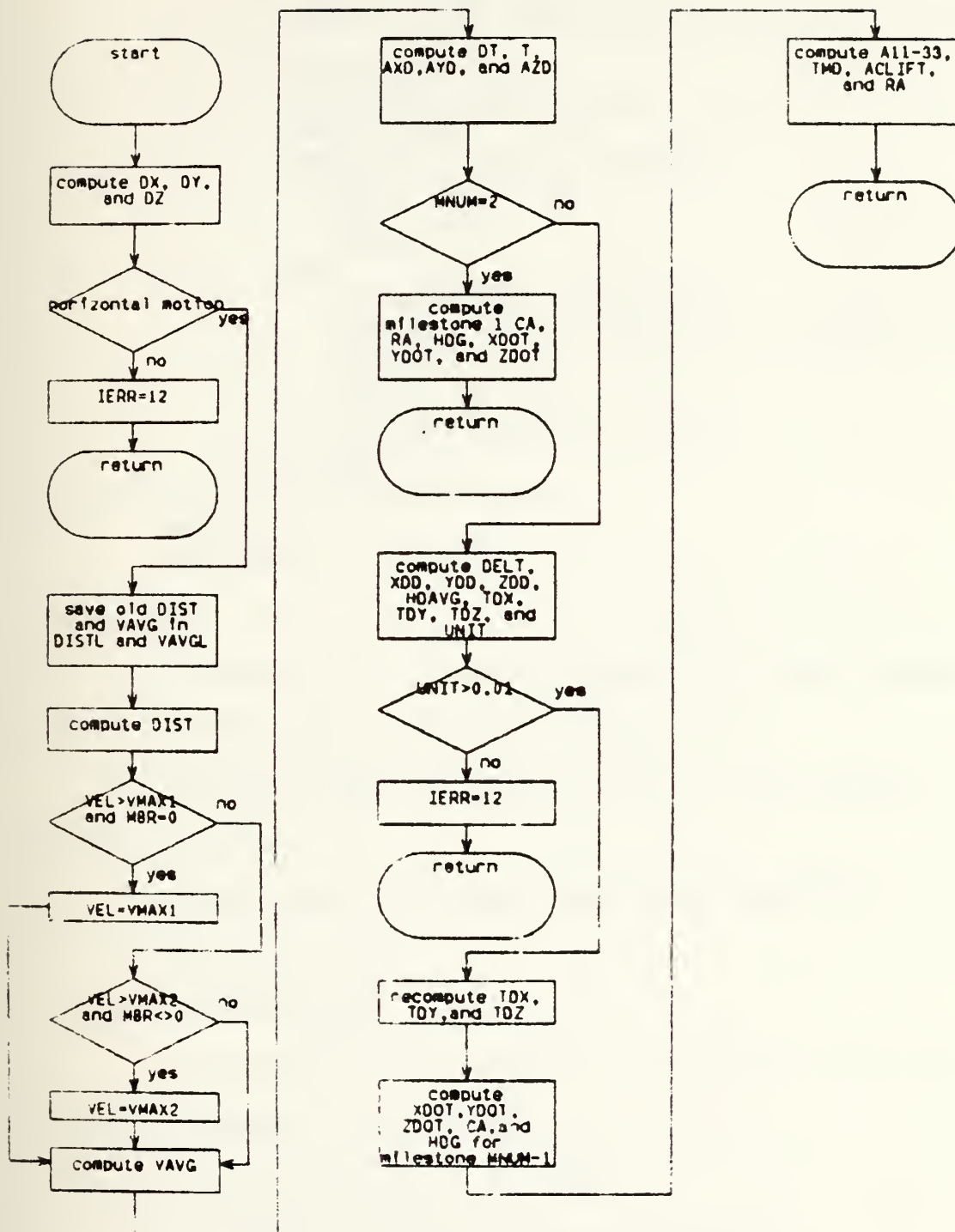


Figure 7. VALSET Flow Chart.


```

    if (VEL > MAX 2 and bomb released) then VEL = VMAX2
    compute VAVG
    compute DT and T
    compute AXD, AYD, and AZD
    if (MNUM = 2) then
        compute CA, RA, HDG, XDOT, YDOT, and ZDOT for milestone 1
    else
        compute DELT
        compute XDD, YDD, and ZDD
        compute HDAVG
        compute TDX, TDY, TDZ, and UNIT
        if (UNIT < 0.01) then
            IERR = 12
            return
        else
            recompute TDX, TDY, and TDZ
            compute XDOT, YDOT, ZDOT, CA, and HDG for milestone
            MNUM-1
            compute A11-A33
            compute TMD, ACLIFT, and RA
            end if [UNIT < 0.01]
        end if [MNUM = 2]
    end if [no horizontal motion]
    return
end VALSET

```

3. ERRCHK

a. Function: This subroutine evaluates the flight parameters to ensure that they meet the desired simulation limits.

b. Parameters Required: IERR--returns the error code for simulation violations.

c. Common Blocks: PAR, PAR1, PAR2, PAR3, PAR4, TAR.

d. Called By: MAIN.

e. Local Variables:

(1) DX, DY--X and Y distances to the target, later used as the X and Y components of the current leg.

(2) DIST--distance to the target.

(3) POPALT--altitude of the pop-up maneuver.

(4) TMSAV--saves the value of TMAX before bombs are dropped.

This is used when the flight path is reset after bombs are dropped.

(5) RHO--density of air in $\text{lb sec}^2/\text{meters}^2 \text{ feet}^2$. The mixture of units is necessary since the coordinate system is in Metric units and wing loading is in English units. This provides the proper units for computing CL and DW.

(6) CL--lift coefficient.

(7) DW--drag divided by weight

(8) TW--thrust divided by weight

(9) DT--time spent on current leg.

(10) TGTHDG--heading from the aircraft to the target in degrees.

(11) ACHDG--direction of the current leg in degrees.

(12) HDGLMT--absolute difference between ACHDG and TFTHDG.

f. Algorithm

```
compute DX, DY, and DIST
if (MNUM = 2 and TMSAV = -1) then
    POPALT = 0
    TMSAV = TMAX
end if
if (MBR = 0 and TMAX <> TMSAV) TMAX = TMSAV [necessary to
reset TMAX when the flight path is reset following a successful
bomb release]
call ERRMK (22)
if (bomb not released) then [MBR = 0]
    if (DIST > POPMIN and Z > APPMAX) then
        IERR = 4
        return
    end if
end if
if (DIST <= POPMIN and Z > POPALT) then POPALT = Z
if (Z < HTMIN) then
    IERR = 3
    return
else if (Z > HTMAX) then
    IERR = 4
```


ERRCHK

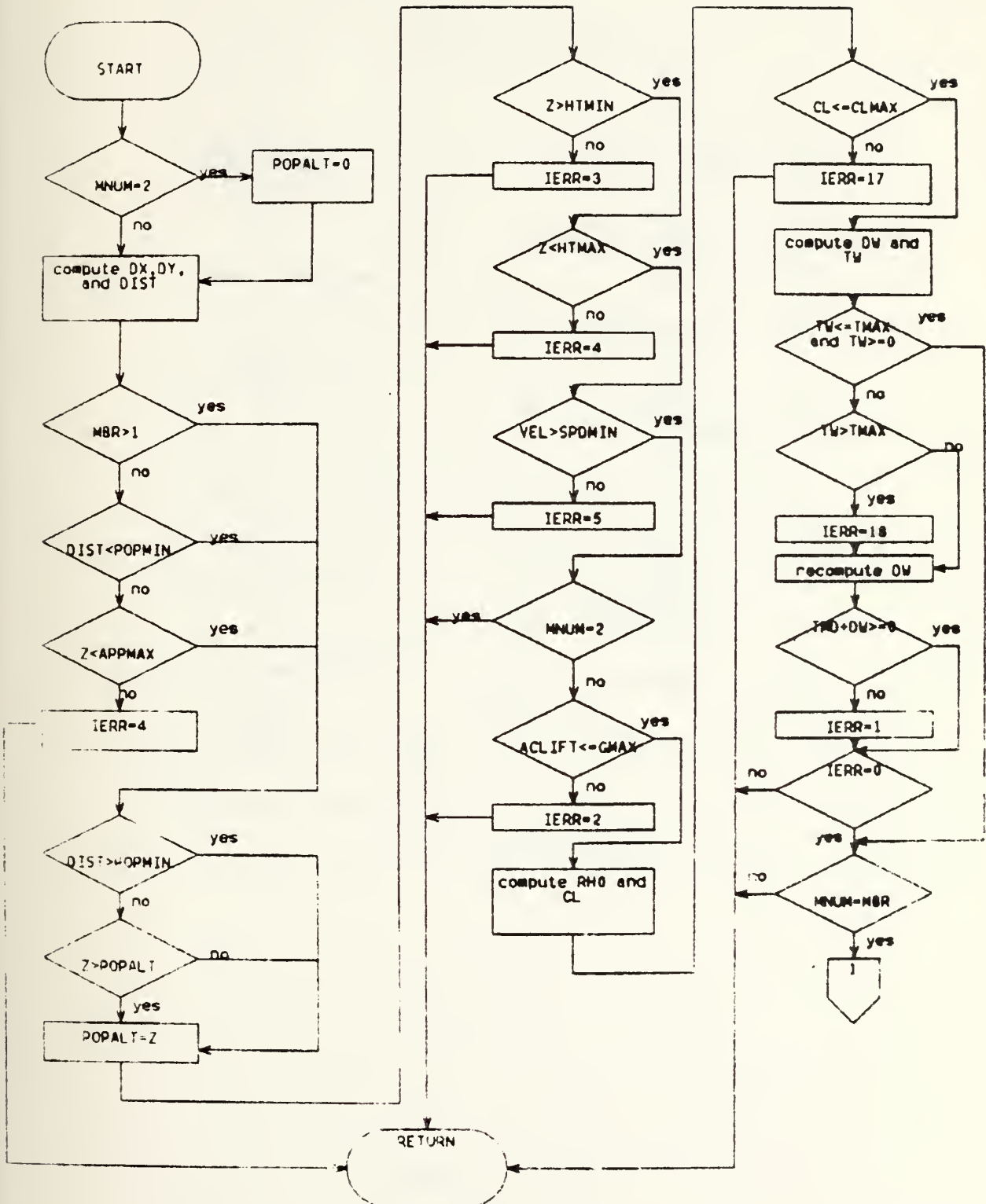


Figure 8. ERRCHK Flow Chart.

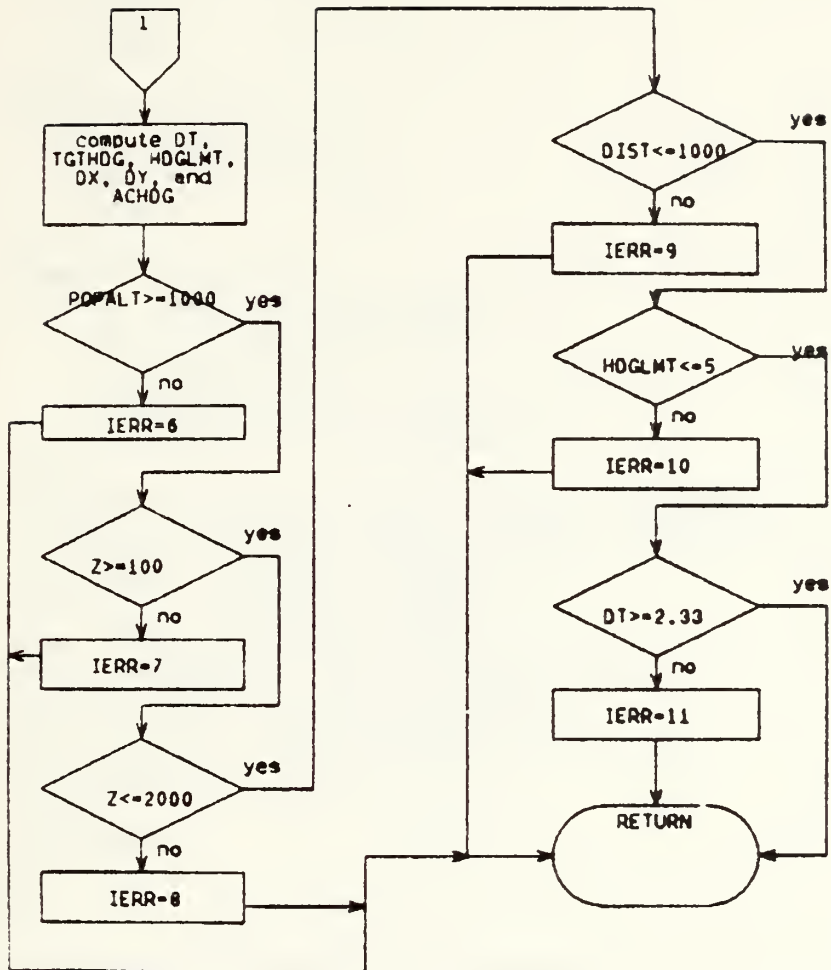


Figure 8 continued.


```

        return
    else if (VEL < SPDMIN) then
        IERR = 5
        return
    end if
    if (MNUM = 2) then Return
    if (ACLIFT > GMAX) then
        IERR = 1
        return
    end if
    compute RHO, CL
    if (CL > CLMAX) then
        IERR = 17
        return
    end if
    compute DW and TW
    if (TW > TMAX or TW < 0) then
        if (TW > TMAX) then IERR = 18
        Recompute DW
        if (TMD + DW <= 0) then IERR = 1
        if (IERR <> 0) then Return
    end if
    if (bomb not dropped this milestone) then Return [MNUM <> MBR]
    compute DT, DX, DY, TGTHDG, ACHDG, and HDGLMT
    if (POPALT < 1,000) then IERR = 6
    else if (Z < 100) then IERR = 7
    else if (Z > 2,000) then IERR = 8
    else if (DIST > 2,000) then IERR = 9
    else if (HDGLMT > 5) then IERR = 10
    else if (DT < 2.33) then IERR = 11
    else TMAX = 1.2*TMAX
    return
end ERRCHK

```

4. PTHPLT

- a. Function: This subroutine notifies the user that his milestone has been accepted and then saves the successful point in a temporary disk file.
- b. Parameters Required: None.
- c. Common Blocks: MN, PAR, PAR1.
- d. Called by: MAIN.
- e. Local Variables: None.

PTHPLT

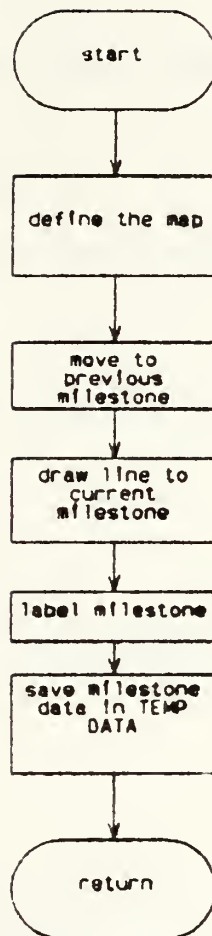


Figure 9. PTHPLT Flow Chart.

f. Algorithm

```
call WIN [define map window]
move to previous milestone
draw line to present milestone
label milestone
write X, Y, Z, VEL, and a 0 to scratch file TEMP DATA
return
end PTHPLT
```

5. SPOT

- a. Function: This subroutine obtains an X/Y coordinate pair and two commands from the user.
- b. Parameters Required:
- (1) X, Y--X, and Y values obtained from the user, returned to the calling routine.
 - (2) L1, L2--two commands obtained from the user, returned to the calling routine.
- c. Common Blocks: None.
- d. Called By: XYZIN.
- e. Local Variables: A, B--dummy variables needed for subroutine VCURSR.

f. Algorithm

```
do until (command <> no) [L2 <> 78]
    input X, Y, and L1
    put point at (X, Y)
    input L2 to verify location
end do
return
end SPOT
```

6. SCENE

- a. Function: This subroutine draws the attack scenario map and associated displays. During reset it obtains the last milestone the user wishes to retain.

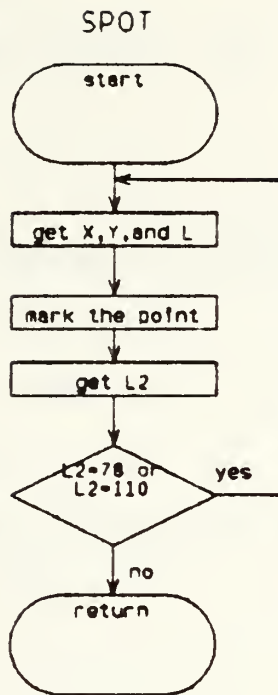


Figure 10. SPOT Flow Chart.

b. Parameters Required:

(1) IRQ--signifies the reset condition: 1 for reset, 0 for initial drawing.

(2) ICT--the last milestone the user wishes to retain, returned to the calling routine.

c. Common Blocks: ERR, MN, TAR, PAR.

d. Called By: MAIN.

e. Local Variables:

(1) MAP--controls whether or not the road, buildings, and river are drawn.

(2) X, Y--X and Y coordinates of points on the map.

(3) IT--loop counter.

(4) IW--scale factor for labelling the axes of the map.

(5) I--index for MKERX.

f. Algorithm

```
clear screen
write village option prompt
read village option [MAP]
if (restart) then [IRQ = 1]
    write restart prompt
    read restart point
end if [restart]
call WIN to define map
if (village drawn) then [MAP = 1]
    rewind disk file PICTUR DATA
    read TX, TY
    do until (TX < -1)
        read TX, TY
        move to TX, TY
        do until TX < 0 and TY > -1)
            read TX, TY
            if (TX > 0) then draw to (TXM TY)
        end do [TX < 0 and TX > -1]
    end do [TX < -1]
end if
call GUNLOC to mark target
```


SCENE

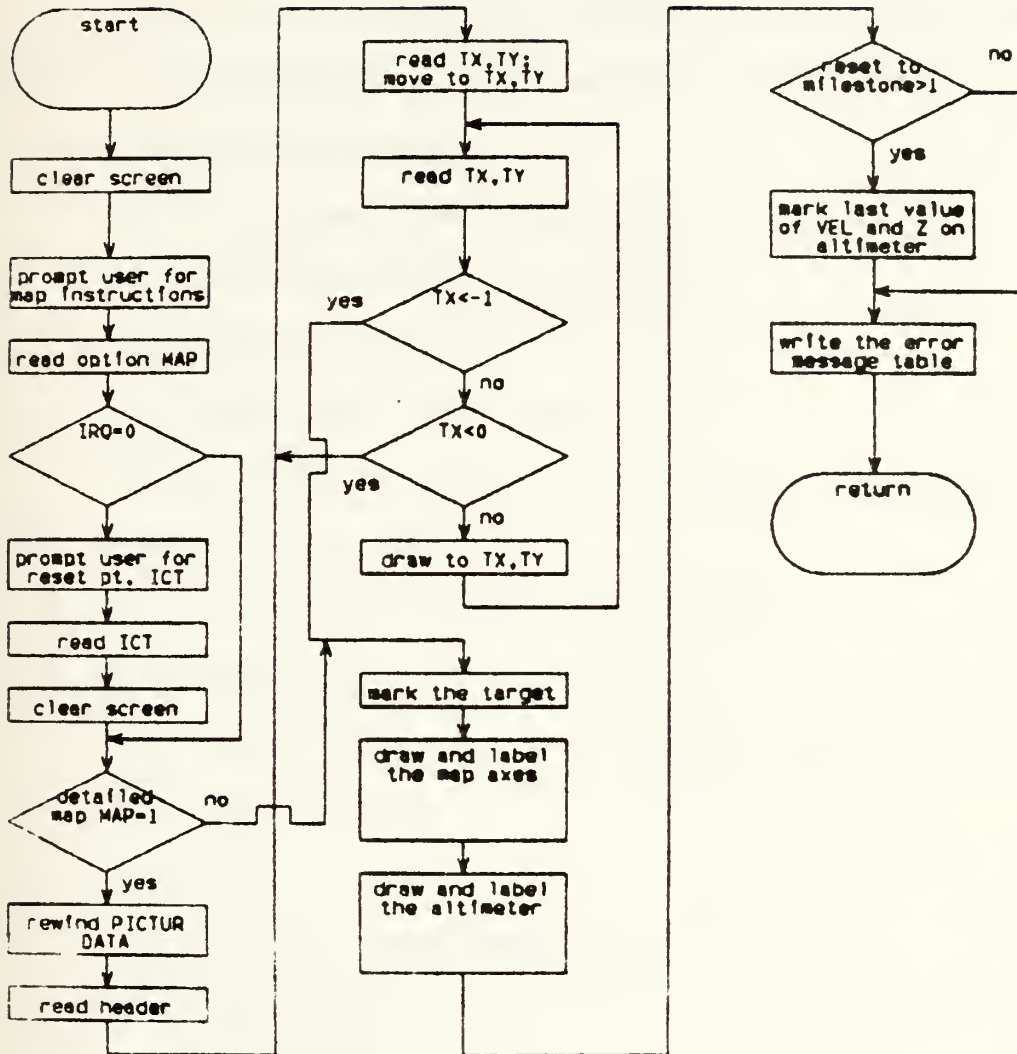


Figure 11. SCENE Flow Chart.


```

label map axes
if (restart) then mark last altitude and velocity
write error messages
return
end SCENE

```

7. WIN

a. Function: This subroutine defines a window on the screen and sets its virtual boundaries. The lower left corner is set to (0,0).

b. Parameters Required:

- (1) LX--minimum screen X coordinate.
- (2) MX--maximum screen X coordinate.
- (3) RX--maximum virtual X coordinate.
- (4) RY--maximum virtual Y coordinate.
- (5) JMP--controls flashing of the window being defined.

c. Common Blocks: MN.

d. Called By: PTHPLT, GUNLOC, SCENE, XYZIN.

e. Local Variables:

- (1) ICT--determines how many times the window will flash.
- (2) I--loop counter.

f. Algorithm

```

define window screen boundaries
define window virtual boundaries
if (flash requested) then
  compute ICT
  do I = 1 to ICT
    flash window
  end do
end if
return
end WIN

```

8. XYZIN

a. Function: This subroutine obtains the X, Y, Z, velocity, and command for the milestone.

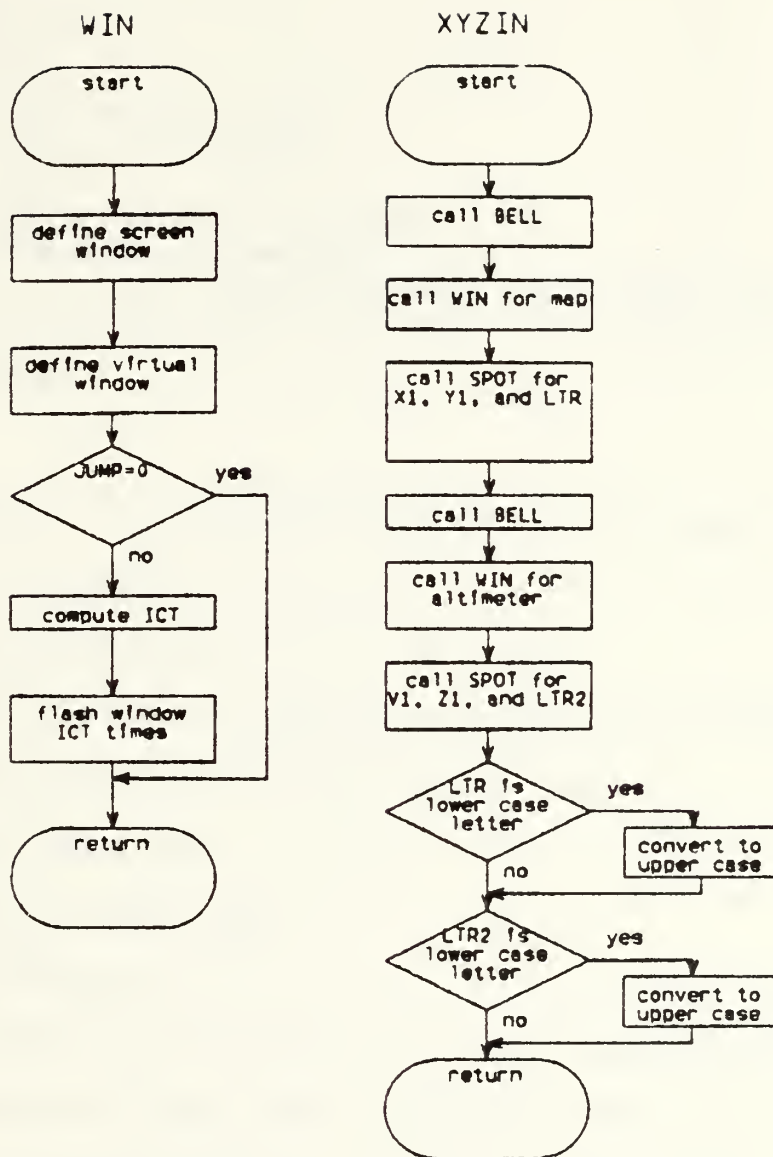


Figure 12. WIN and XYZIN Flow Charts.

b. Parameters Required: None.

c. Common Blocks: MN, LOC.

d. Called By: MAIN.

e. Local Variables: None.

f. Algorithm

```
call BELL
call WIN to define the map
call SPOT for X, Y, LTR
call BELL
call WIN to define the altimeter
call SPOT for Z, V1, and LTR2
if (LTR is lower case) then convert LTR to upper case
if (LTR2 is lower case) then convert LTR2 to upper case
return
end XYZIN
```

9. GUNLOC

a. Function: This subroutine marks the indicated spot with a "+" and draws a circle around it.

b. Parameters Required:

(1) GX, GY--X and Y coordinate of the point to be marked.

(2) RAD--radius of the circle marking location.

c. Common Blocks: MN.

d. Called By: MAIN, SCENE.

e. Local Variables:

(1) ISTEP--size of the step in the loop which draws the circle. Small circles use fewer points, therefore, ISTEP will be larger.

(2) Angle--angle around the circle at which a point will be drawn.

(3) DX, DY--X and Y coordinates of the point to be drawn.

f. Algorithm

```
call WIN [define map window]
draw cross at site
```


GUNLOC

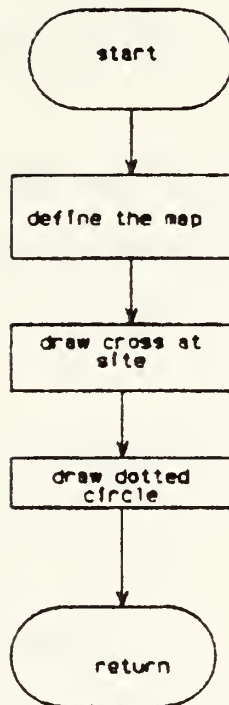


Figure 13. GUNLOC Flow Chart.


```

compute ISTEP
draw dotted circle around site
return
end GUNLOC

```

10. BEGIN

- a. Function: This subroutine initializes the graphics and requests the user's options.
- b. Parameters Required: None.
- c. Common Blocks: MN, PAR, OPT.
- d. Called By: MAIN.
- e. Local Variables: KON--user input: 0 indicates the user wishes to use default simulation parameters, any other value causes a call to CONCHG to allow the user to input his own parameters.
- f. Algorithm

```

call INIT [initialize graphics]
clear screen
write gun option prompt
read IGUN
write milestone option prompt
read IMP
write error option prompt
read KER
write terminal option prompt
read IBAUD
write parameter input prompt
read KON
if (KON <> 0) then call CONCHG
set IPLOT = 1
compute map and altimeter boundaries
return
end BEGIN

```

11. GUNCHK

- a. Function: This subroutine checks gun locations against emplacement rules.
- b. Parameters Required:
 - (1) X, Y--location of the gun site.

BEGIN

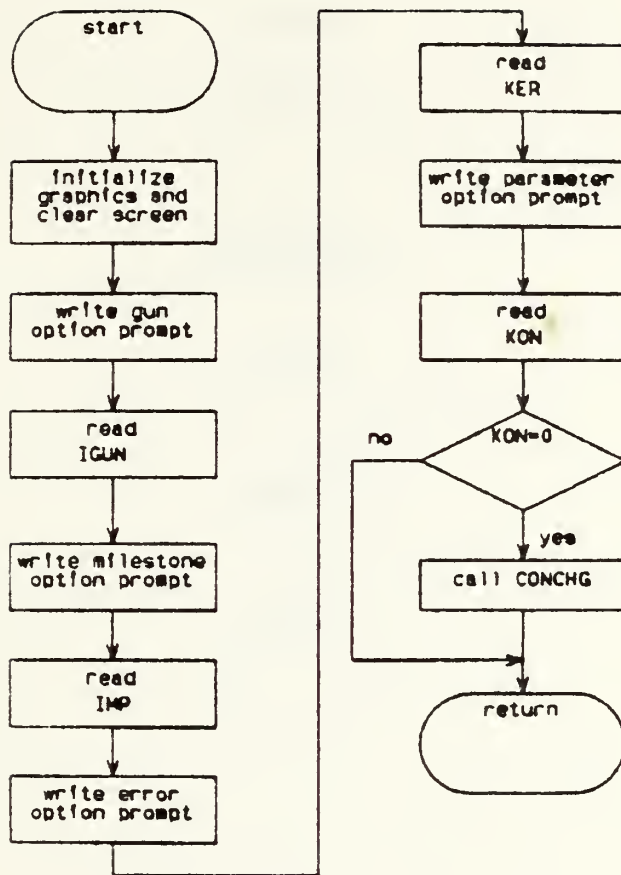


Figure 14. BEGIN Flow Chart.

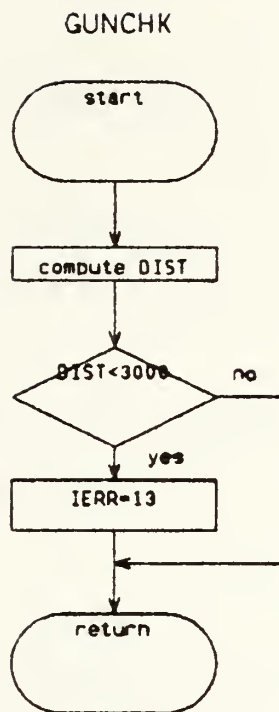


Figure 15. GUNCHK Flow Chart.

(2) IERR--error code set to 13 if emplacement rules are violated.

c. Common Blocks: TAR.

d. Called By: MAIN.

e. Local Variables: DIST--distance from the gun site to the target.

f. Algorithm

```
compute DIST
if (DIST < 3,000) then IERR = 13
return
end GUNCHK
```

12. PRESET

a. Function: This subroutine writes the P001 and MICE II files for the user.

b. Parameters Required: PAR, PAR1, PAR2, OPT.

c. Common Blocks: None.

d. Called By: None.

e. Local Variables:

(1) RCSTAB--radar cross-section table.

(2) VAT1N2--vulnerability table against type 1 and 2 weapons.

(3) VAT3--vulnerability table against type 3 weapons.

(4) VAT5--vulnerability table against type 5 weapons.

(5) TINC--total flight time, T(MNUM), divided by 1,000

with 0.0008 added to eliminate round-off errors due to P001's method of time calculation.

(6) TINKI--temporary time marker.

(7) TINK--array with nine elements, the Ith element has a value $I * T(MNUM) / 10$.

(8) FTFAC--conversion factor from meters to feet, set to 3.28084.

PRESET

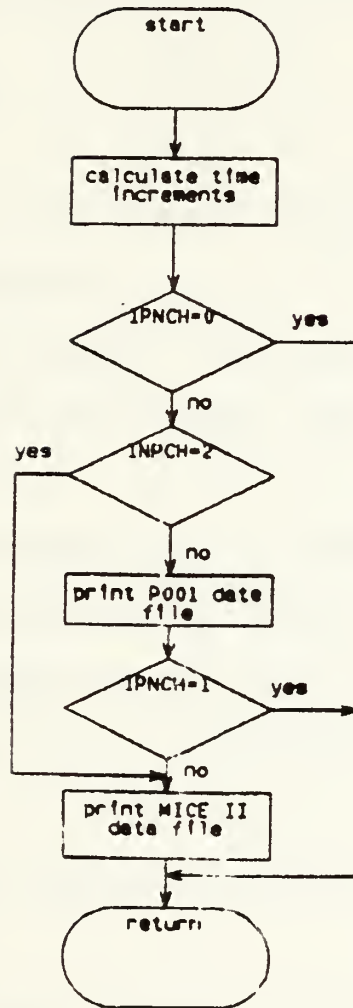


Figure 16. PRESET Flow Chart.

(9) XFSAM, YFSAM, ZFSAM--X, Y, and Z coordinates, in feet, of the missile site.

f. Algorithm

```
compute TINC AND TINK
if (P001 file requested) then write P001 DATA file [IPNCH = 1 or 3]
if (MICE file requested) then [IPNCH = 2 or 3]
    print MICE DATA file header
    convert X, Y, Z, and VEL from meters to feet
    print remaining data to MICE DATA
end if
return
end PRESET
```

13. ELFIN

a. Function: This subroutine requests the user's output options and closes the graphic routines.

b. Parameters Required: LAST--this is both a command parameter to ELFIN and a returned value to MAIN. Sent as a command, it has the following meaning:

(1) 0--no further action is required.

(2) 1--ask the operator whether or not he is finished with the flight path or wishes to continue at a later time.

If LAST was sent as 0, it is returned as an 83. If a 1 was sent, it returns as follows:

(1) 1--user wishes to continue the flight path at a later time.

(2) 83--user is finished with the flight path.

c. Common Blocks: MN, OPT.

d. Called By: MAIN.

e. Local Variables: ICT-- loop counter that determines how often the prompt line will be flashed.

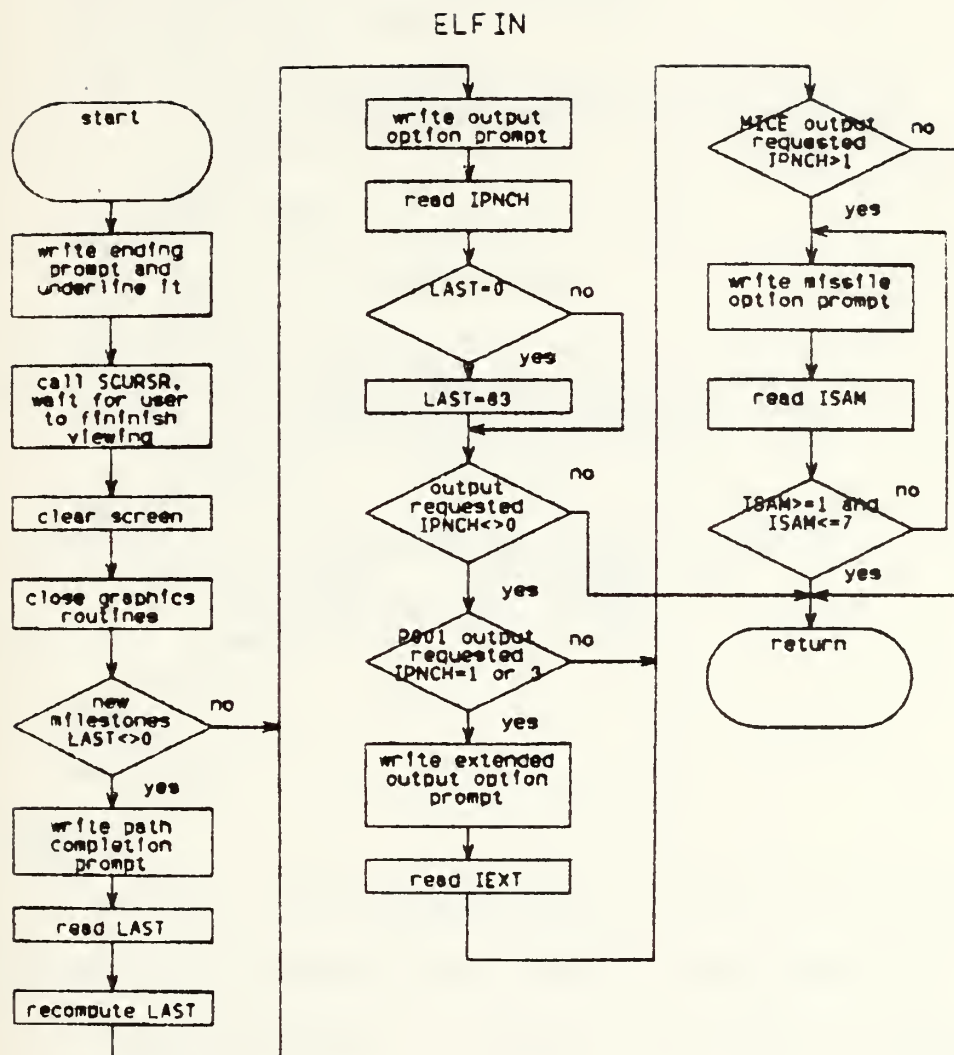


Figure 17. ELFIN Flow Chart.

f. Algorithm

```
write ending prompt
compute ICT
do I = 1 to ICT
    call BELL
    underline prompt
end do
call SCURSR [wait for user to finish viewing screen]
clear screen
call FIN [closes graphics routines]
if (LAST <> 0)
    do until (ABS (LAST) < 2)
        write flight path completion option prompt
        read LAST
    end do
    compute LAST
end if
write output file option prompt
read IPNCH
if (LAST = 0) LAST = 83
if (no output requested) then return [IPNCH = 0]
if (POO1 file requested) then [IPNCH = 1 or 3]
    write extended output option prompt
    read IEXT
end if
if (MICE file requested) then [IPNCH = 2 or 3]
    do until (ISAM > = 1 and ISAM < = 7)
        write missile option prompt
        read ISAM
    end do
end if
return
end ELFIN
```

14. AIMPT

- a. Function: This subroutine marks a line from the aircraft's current position to the target to help align the bomb release leg.
- b. Parameters Required: I--number of the current milestone.
- c. Common Blocks: PAR, TAR.
- d. Called By: MAIN.
- e. Local Variables: None.

AIMPT

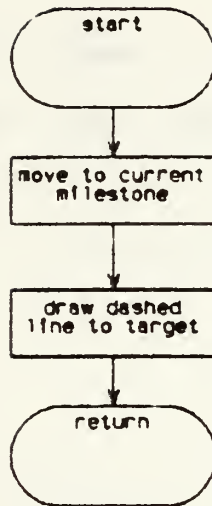


Figure 18. AIMPT Flow Chart.

f. Algorithm

```
move to (X(I), Y(I))
draw dashed lined to target
return
end AIMPT
```

15. CONCHG

a. Function: This subroutine allows the operator to change the simulation parameters.

b. Parameters Required: None.

c. Common Blocks: PAR3, PAR4, TAR.

d. Called By: BEGIN.

e. Local Variables: K--counter to indicate which parameter the user wants to change.

f. Algorithm

```
clear screen
do until (input completed) [K = 1]
  do until (K >= 1 and K <= 17)
    write prompt
    read K
  end do
  if (K > 1 and K < 16) then
    write prompt for parameter to be changed
    read new value of the parameter
  else if (K = 16)
    read parameter values form disk file PAR SAV
  else if (K = 17)
    list parameter values to the terminal
  end if [K > 1 and K < 16]
end do [input completed (K = 1)]
write parameter values to disk file PAR SAV
return
end CONCHG
```

16. ERRMK

a. Function: This subroutine sends prompts and error messages to the user.

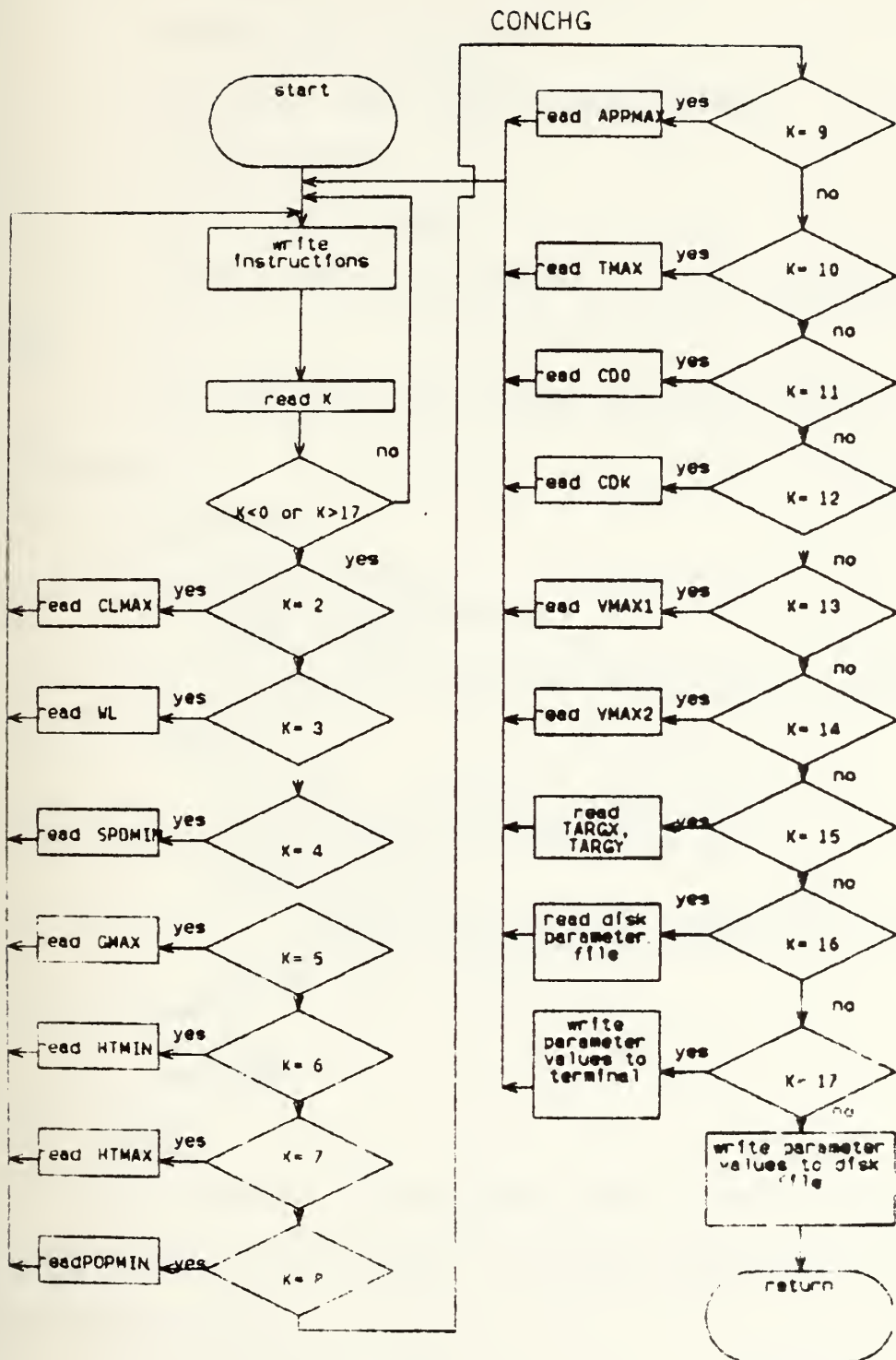


Figure 19. CONCHG Flow Chart.

b. Parameters Required: IERR--this indicates which message is to be sent to the operator.

c. Common Blocks: ERR, MN.

d. Called By: MAIN, VALSET, ERRCHK, SAMCHK.

e. Local Variables:

(1) J--column index.

(2) MKX, MKY--working variables to help define the message line.

(3) K--counter which defines how many times the prompt line is displayed.

f. Algorithm

```
if (IERR > 20) then
  compute MKX and MKY
  do I = 1 to 6
    draw line from (MKX, MKY) to MKX + 30, MKY)
  end do
else
  J = 2
  if (IERR <= 6) then J = 1
  else if (IERR > 12) then J = 3
  compute K and MKY
  do I = 1 to K
    draw line from (MKERX(J), MKERY(IERR) to (MKX, MKERY(IERR))
    if (IERR < 10) call BELL
  end do
end if
return
end ERRNK
```

17. SAMCHK

a. Function: This subroutine checks the missile location for emplacement rule violations. A violation causes a call to ERRMK with an error code of 19.

b. Parameters Required:

(1) X--X coordinate of the missile site.

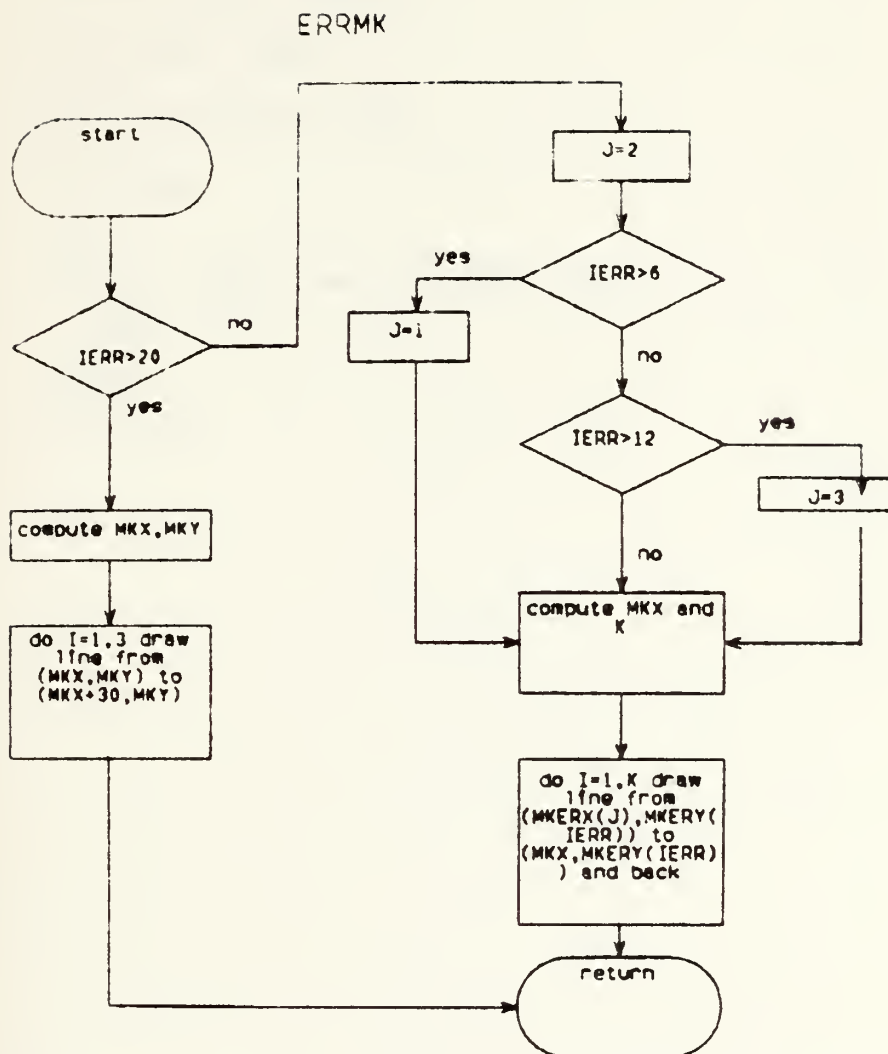


Figure 20. ERRMK Flow Chart.

(2) IERR--error code, set to 0 if no violation occurs and set to 1 if it does.

c. Common Blocks: None.

d. Called By: MAIN.

e. Local Variables: None.

f. Algorithm

```
IERR =  
if (X < 6,000) then call ERRMK (19)  
    IERR = 1  
end if  
return  
end SAMCHK
```


SAMCHK

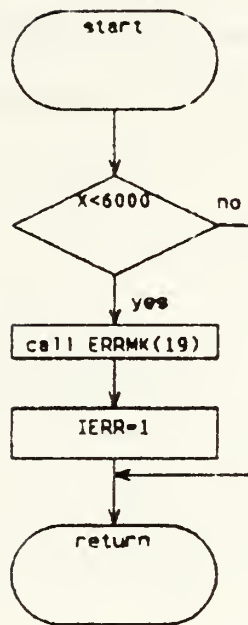


Figure 21. SAMCHK Flow Chart.

VII. MODIFICATION INSTRUCTIONS

A. GENERAL

As stated in the introduction, one goal of this effort was to develop a program that would be transportable to other systems. This section will discuss the general requirements to achieve this conversion and then describe how it can be implemented in the two versions that currently exist at the Naval Postgraduate School.

The first change that must be made is the conversion to the user's graphics system. The program requires the following capabilities:

1. Screen and Virtual Window Definition

TWINDO defines a window in screen coordinates while DWINDO defines the virtual coordinates the current screen window will assume. NEWPAG clears the screen, but does not change the window definitions.

2. Move and Draw

A "move" positions the electron beam at a point on the screen, but does not draw anything. A "draw" creates a straight line on the screen from the current beam position to the position identified in the parameter list. MOVABS and DRWABS are absolute screen coordinate move and draw commands. MOVEA and DRAWA are the virtual coordinate commands to accomplish these functions. MOVREL is a virtual move relative to the current beam position and DASHA is a virtual draw that creates a dashed line.

3. Print to the Graphics Screen

Printing to the screen creates alphanumeric characters on the screen, as opposed to connected lines. ANMODE enables the printing capability.

4. Cursor Input

Cursor input allows the user to enter a command value and the X and Y coordinates of a point on the screen. VCURSR accepts data in virtual coordinates while SCURSR works in screen coordinates.

The user's functions for these four actions must be substituted for the PLOT-10 functions according to Table 1.

The next change that must be made is to implement the user's aircraft performance constraints. The module VALSET contains the equations for computing the necessary performance values and has the ability to indicate boundary errors via the parameter IERR. The constraints on the system are contained in module ERRCHK, and performance violations are passed with the parameter IERR. The various values assigned to IERR are used by ERRMK to communicate with the user whenever the graphics routines are being used. Therefore, not only the error messages, but also user prompts must be supported by this module. There is one other constraint on the value assigned to IERR. When assigned a value of 12, the user's option to ignore errors is overridden. This allows the programmer to guard against situations the program is unable to handle, such as division by zero.

The final change is in the format of the output files. Because the program was written to support a particular Naval Postgraduate School course, a fixed number and type of weapons is expected. The module MAIN

TABLE 1
MODULE GRAPHICS REQUIREMENTS

	TWINDO	VWINDO	NEWPAG	MOVABS	DRWABS	MOVEA	DRAWA	MOVREL	DASHA	ANMODE	VCURSR	SCURSR
ERRMK				X	X							
PTHPLT						X	X			X		
GUNLOC						X	X					
SPOT											X	
SCENE			X	X		X	X	X		X		
WIN	X	X		X	X							
BEGIN			X									
ELFIN			X	X	X							X
AIMPT						X			X			
CONCHG			X									

accepts the input of the first six guns and one SAM site and assigns a seventh gun location. The module PRESET assigns the guns the following description:

GUN	TYPE	MODE
1-2	1	1
3-4	2	1
5	3	4
6	3	3
7	5	3

To change this assignment, the module MAIN must be modified to accept and display the additional information and PRESET must be rewritten to recognize varying numbers and types of weapon sites.

B. THE KEYBOARD VERSION (KBPIP)

The keyboard version was written to operate on any keyboard terminal. To a large extent, this was simply a matter of removing the graphics functions from GRPIP. Modules that had only graphics functions were not eliminated, but were converted into stubs. The remainder of this section will describe the changes to the modules in KBPIP that are necessary for conversion.

1. MAIN

This module had to notify module XYZIN whether it was expecting weapon or milestone information. A parameter was added to XYZIN, with an 0 indicating weapon input and a 1 for milestone data.

2. ERRMK

This module contains messages that were previously written on the screen by SCENE. It uses FORTRAN WRITE statements to notify the

user, rather than highlighting a previously written message. In addition, the input data is echo printed to assist the user, since this function was automatic in the graphic version.

3. PTHPLT

This module now writes a message indicating that the milestone has been accepted as valid.

4. GUNLOC, WIN

These modules are now simply stubs.

5. SPOT

This module now uses READ and WRITE rather than cursor inputs.

This means that it must be able to identify what type of data is expected in order to write the appropriate prompts. The parameter L1 is sent with a value of 0 to indicate that X and Y are to be read, and a value of 1 indicates that the value of L2 must be checked. If L2 is 0, the calling routine expects only an altitude. If it is 1, the calling routine expects an altitude, velocity, and command. The module must then convert the command from the user's format to that used by MAIN.

6. SCENE

The entire map drawing function has been eliminated. When called during reset, the module prints the data for the last milestone retained by the user, to assist in the selection of the values for subsequent points.

7. XYZIN

The module is sent the parameter IV. It uses the value of this parameter to determine whether the calling routine needs only X, Y, and Z coordinates, or X, Y, Z coordinates, velocity data, and a command. It passes this information to SPOT via parameters LTR, LTR2, and LDM.

8. BEGIN

The initialization of graphics routines has been eliminated.

9. AIMPT

This module now gives the distance to the target and how much the current X and Y values must be changed to align the flight path correctly.

10. ELFIN

The release of the graphics resources has been eliminated.

11. CONCHG

The screen clearing command is no longer required.

C. THE IBM GRAPHICS VERSION (IBMPIP)

The IBM graphics version, with its dual screens, was to a certain extent a merger of the PLOT-10 and keyboard versions. The necessary changes are described below.

1. PTHPLT, GUNLOC, SCENE, WIN, ELFIN, AIMPT, SPOT

IBM functions are substituted for equivalent PLOT-10 functions.

2. BEGIN

New screen coordinates are used. In addition, the graphics initialization and screen erasing functions are moved from the beginning to the end of the module.

3. CONCHG

The screen clearing function is eliminated.

4. ERRMK

The module flashes a small window on the graphics screen to indicate a message is on the alphanumeric screen. It uses the

alphanumeric screen to write the message. In addition to the message, the data input on the most recent attempt is printed if an error occurs.

VIII. MAP GENERATION

To support the graphics capability of the program, it was necessary to provide a map of the attack area to the user. To do this, it was necessary to select points on the actual map and copy them to a data file for later use. This can be done manually with rulers and pencil, which is subject to all the problems involved in properly marking and transferring the data. An alternative was to utilize the graphics capability of the system to generate the map information. To accomplish this, a program called SCENE was written.

The prerequisite to utilizing SCENE is a transparency of the desired map. SCENE expects a 6 x 4 inch transparency representing an 18,000 x 12,000 unit field. This can be changed by changing the values of MINX, MAXX, MINY and MAXY and the values in the DWINDO call.

The program starts by asking the user whether he wants to create a new map or continue an old one. A positive integer means a new map, zero or a negative number indicates an old map. If selected, the old map is then drawn. The user then places the transparency over the map outline drawn on the screen. Points are input in the following manner. First, the cursor is positioned under the selected point. A key is pressed to mark the spot. The capital "M" and "S" keys have a particular meaning. The "M" key informs the program that the following call will be a move, not a draw. The "S" key marks the final spot and informs the program to stop execution. All other keys indicate that the next call is a draw command.

The data is written to a FORTRAN data set 9 using 2F10.2 format. The data has the following meanings:

1. Two positive values--these represent a data point on the map.
2. -0.5, 0.0--this is a mark to indicate that a move to the coordinates in the next entry is to occur.
3. -2.0, 0.0--this is the end of file marker. If the user is continuing work on a map, it will be necessary to erase the original marker after he finishes the current session. This is done by editing the data file and removing the first end of file marker.

In addition to the cursor input, data points can be entered by editing the file. This allows the user to smooth curves by adding intermediate points and to make corners connect exactly. One example of these capabilities is the drawing of the buildings. Rather than trying to exactly line up the corners, only the diagonal defining the rectangle was drawn with the cursor. The data file was then edited to fill in the missing corners. This ensured that the corners were perpendicular and met at the line end points.

IX. PROBLEMS AND IMPROVEMENTS

Two major problem areas were encountered in terms of software development. The first involved the utilization of graphics resource drivers and the second was version control. A major problem with the PLOT-10 graphics package is that there are errors in the Naval Postgraduate School's implementation of several of the translation tables. As a result, the system will occasionally become highly erratic; drawing random lines over the entire screen, or reading incorrect values from the cursor. This erratic behavior was the original reason the verification input was included in the graphics sequence. This problem remains uncorrected.

The major difficulty with the IBM graphics package was the lack of familiarity with the package on the part of computer center personnel. The package was installed in April 1982, and the only documentation available was the IBM technical manual. Implementing IBMPIP was, therefore, limited to using those functions which could be easily identified, primarily the bridge functions. Therefore, it was not possible to utilize the extensive capabilities of the package. As more experimentation with the package is afforded, the capabilities of this package should be enhanced. In addition, there are plans to obtain a core graphics system which will support TEKTRONIX, IBM, and VERSATEC plotting routines in a device-independent manner. This means that the user will be able to use high-level graphic functions. These will execute equally well on all three systems and the device-specific calls will be generated

by the high-level functions at execution time. If the programmer wishes to use the unique capabilities of a particular device, such as cursor redefinition in the IBM package, he will still have to use the functions specifically dedicated to that device.

Since the core system was not available, the two graphics systems required separate software versions to execute on their respective hardware. The non-graphics keyboard implementation required a third version. This caused a significant management problem. What was a simple change on one system was frequently difficult on another. For instance, correction of errors when milestones were read from the disk was quite simple to accomplish with the keyboard version. The prompt was displayed, the response was accepted, and the appropriate action was carried out. Once this was tested, the source statements were copied into IBMPIP and checked out. This, too, went well. However, the change to GRPIP was extremely difficult. First, the message had to be added to the message list in SCENE, as opposed to ERRMK in the other two versions. ERRMK had to be changed to accommodate a new error location. Finally, the location of the various error messages had to be rearranged four times before one was found that did not result in the new message overwriting another message. Similar problems occurred when converting graphic to non-graphic techniques. Although the functions are identical in each version, most changes in the program resulted in three similar, but unique, changes in the implementation. When the time required to recompile and test these versions is considered, a simple change for cosmetic effects rapidly loses its desirability.

Although not as difficult an obstacle as the software, the hardware did pose a problem. The capabilities of a storage tube device severely limited the alphanumeric interaction with the user. The first problem arose with GRPIP, which was how to write messages to the user without destroying the graphic display or overwriting a previous message. The solution was to write all possible messages beforehand and then identify their location on the screen for later use. This in turn meant a trade-off between a display large enough to be useful while retaining sufficient working space for writing the messages. With the availability of the IBM package, this problem was significantly reduced. With the second screen, alphanumeric interaction was relatively simple. The problem then became one of how to notify the user to change input screens. This was solved by adding an attention window, which flashed to inform the user that something was occurring on the other screen.

There are several improvements that can be made to increase the program's utility. A major improvement would be to use the data input by the user to derive several intermediate milestones. A quick way to do this would be to use the averages already computed to create additional milestones. A better way would use the operator's input and computed averages to fit a curve to the points if desired by the user. Additional milestones would then be created by selecting intermediate points on the curve. In addition to smoothing the flight path, realism could be enhanced by providing for terrain masking. P001 provides for gun masking arcs and if the terrain was identified as geometric shapes, chopped conic mountains and a polynomial river for example, these arcs could be computed.

In addition to increased realism, the program could be improved by adding flexibility to the weapon and aircraft descriptions. The present program utilizes a fixed number of weapons firing at a particular type of aircraft. This was necessary to limit the action required by the students using the program. Making the number and type of weapons variable would require a relatively modest effort, but would greatly expand the software's flexibility. In addition, it would improve the realism of the scenario. P001 only supports one firing arc per gun. By making the number of guns variable, a single gun site which has several distinct firing arcs could be identified as several guns at one location, each with different firing arcs. The aircraft type can be changed by changing the vulnerability and radar cross-section data statements. Providing access to external files to obtain this information would also expand the flexibility of the program.

APPENDIX A
USER'S MANUAL

AE 3251
AIRCRAFT COMBAT SURVIVABILITY

A STUDY
of
AIRCRAFT ATTRITION
in a
HOSTILE AAA AND SAM ENVIRONMENT

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

SPRING QTR, 1982

I. INTRODUCTION

This aircraft attrition study is designed to present the student with an opportunity to see first-hand how the survivability of an aircraft can be evaluated in a given combat scenario. The methods employed in this study are those used by both industry and government when making decisions in the survivability analysis and design of an aircraft weapon system. In this study, one computer program named P001 (the AFATL Antiaircraft Artillery Simulation Computer Program) will be used to simulate the flight of a typical Naval attack aircraft through a hostile antiaircraft artillery (AAA) environment and to compute the aircraft probability of survival. Another computer program, MICE II (the Missile Intercept Capability Evaluation Program), will be used to compute the survivability of the aircraft on the same flight path against a typical short- to medium-range surface-to-air missile (SAM).¹

¹The DoD specifies the use of P001 and MICE in all nonnuclear survivability assessments in MIL-STD-2069, REQUIREMENTS FOR AIRCRAFT NONNUCLEAR SURVIVABILITY PROGRAM, 24 August 1981.

II. PROBLEM DEFINITION

- A. You are going to conduct a survivability assessment of a familiar Naval aircraft, shown in Figs. 1 and 2, on a typical attack mission to destroy the bridge shown in Fig. 3.
- B. The class will be divided into groups of four, with two members in each group on the blue team and two members on the red team.
- C. Each team will use P001 and MICE II to determine the survivability of the aircraft in the class problem scenario, as follows:
 - 1. Each team will select a flight path to the bridge according to the rules of the scenario given in Section IV. Keep this path a secret.
 - 2. Each team will also select the locations of six AAA emplacements and one SAM that will defend the bridge against an air attack. Locate the weapons according to the order of battle given in Section IV. Keep these locations secret, also.
 - 3. Each team will conduct an attack against the other team in the group.
 - 4. The input data file for the computer runs for the blue team attacking the bridge defended by the red team will consist of the flight path of the blue aircraft flying through the AAA and SAM emplacements selected by the red team.
 - 5. Conversely, the input data file for the computer runs by the red team against the blue team will consist of the flight path of

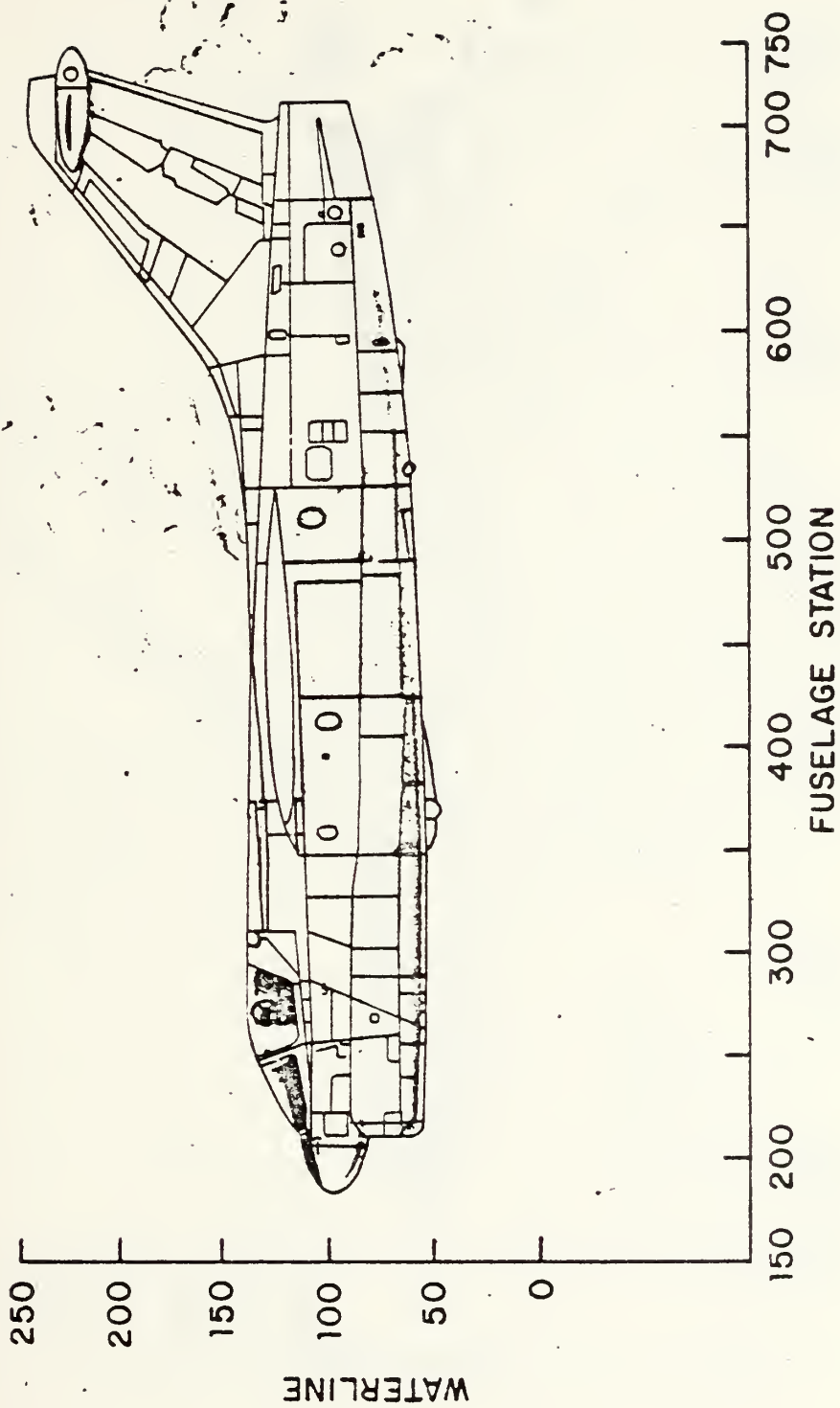


Figure 1

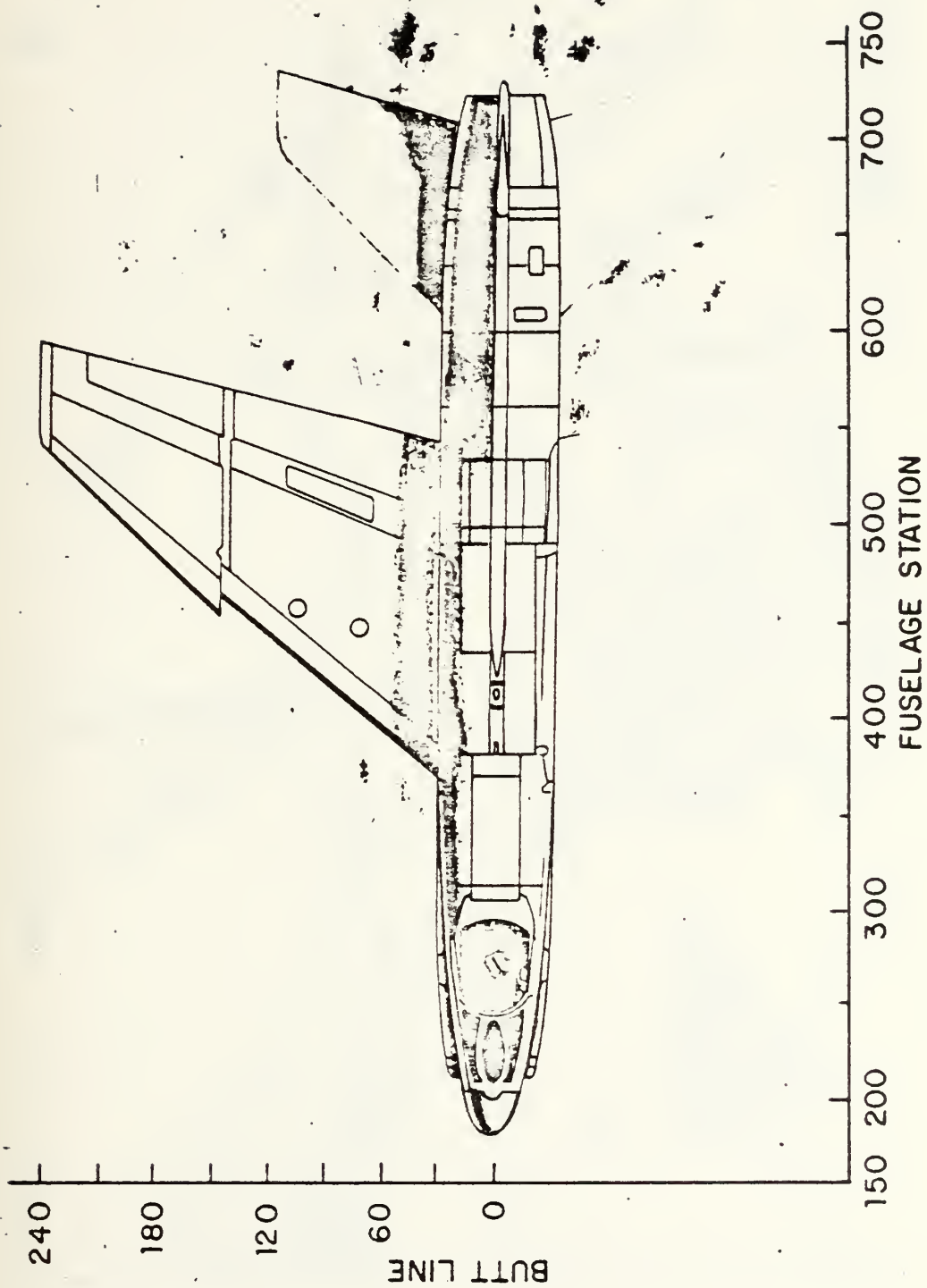


Figure 2

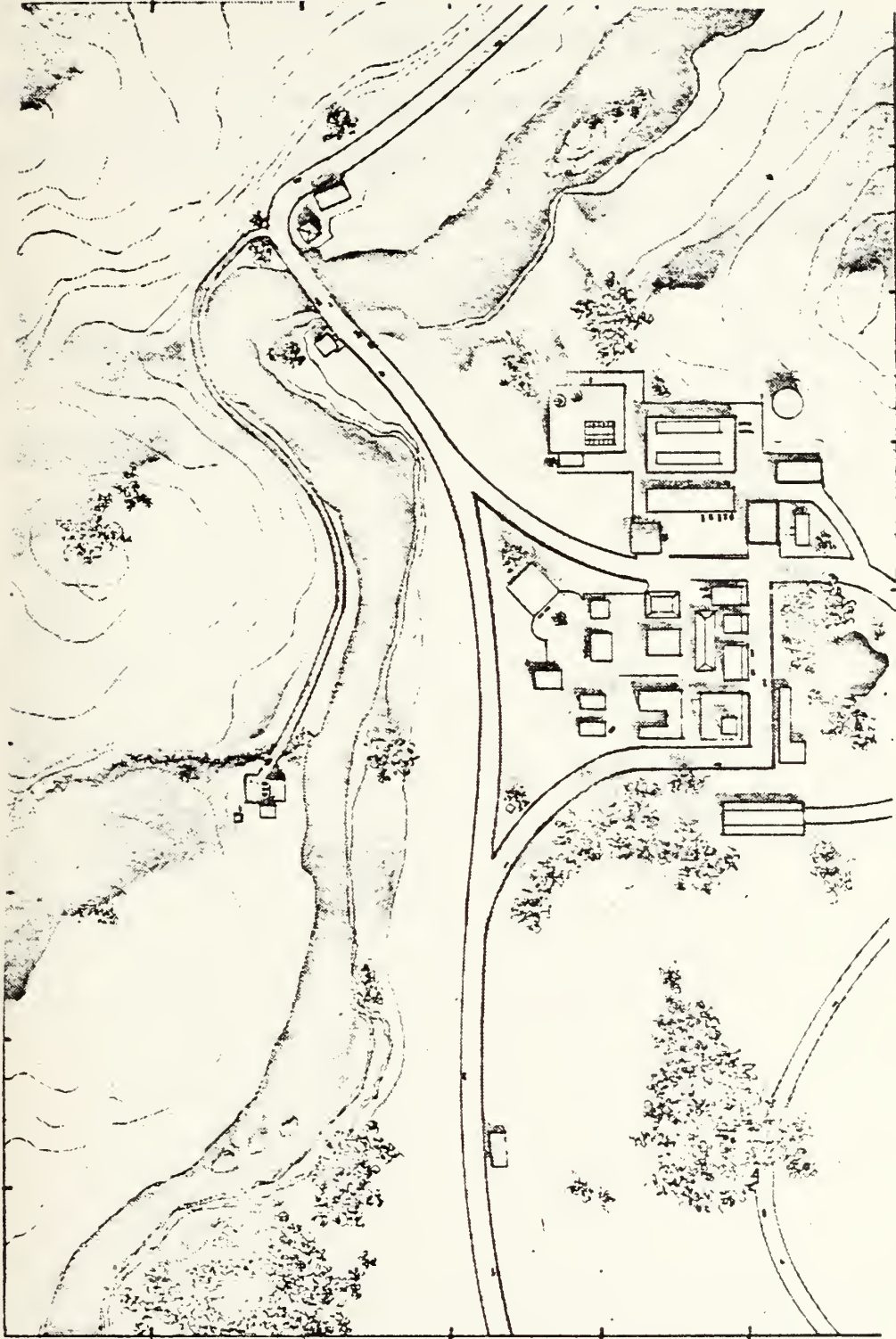


Figure 3

the red aircraft flying through the AAA and SAM emplacements selected by the blue team.

- D. May the best team win. A small prize will be awarded to the team whose aircraft has the highest probability of survival against their opponent's weapon distribution.

III. SCENARIO DESCRIPTION

A. This scenario is purely for instructional purposes and is not based on any actual or planned combat attack situation. The target site, order of battle, and flight path and weapon delivery parameter limits have been chosen only to provide guidelines for the class problem. As much realism has been introduced for the players as possible while retaining an unclassified scenario.

B. Your target is the bridge shown in Fig. 3 located at:

X: 14,000 meters

Y: 7,220 meters

Z: 20 meters

Heavy military supply traffic has been reported in this area.

Your mission is to destroy this vital supply link.

C. The following order of battle has been gathered from intelligence reports of the target area:

SAM - one site in the vicinity of the target

AAA - two type 1 mode 1

two type 2 mode 1

one type 3 mode 4

one type 3 mode 3

one type 5 mode 3

(NOTE: Gun types and their relationship to AAA and the SAM type will be discussed in class.)

- D. The SAM threat requires that the inbound approach to the target be made from the west at low level. A pop-up maneuver is required to visually identify the target followed by a dive bombing run to weapon delivery. The aircraft ordnance consists of MK82 500-pound Snakeye bombs. Egress must be made to either the north or south, depending on individual strategy.
- E. The following is a list of scenario limitations to be used in the development of your strategy:
1. Aircraft cruise speed--90 to 220 meters per second (400 to 500 knots).
 2. Inbound altitude--70 to 450 meters.
 3. Maneuvering--the aircraft is limited by its thrust, speed brakes, and loading. A maximum speed of approximately 225 meters/sec is attainable before bomb release, and it increases to about 235 after weapon release. The thrust available and speed brakes limit the amount of velocity can be changed. Combined with the loading, they serve to limit the amount of turn allowed. In both cases, the longer the leg, the more change can be accepted.
 4. Pop-up maneuver
 - a. Commence maneuver--a maximum of 6,000 meters from the target.
 - b. Maneuver altitude--in order to locate the target, you must pop-up to a minimum altitude of 1,000 meters.
 5. Ordinance delivery
 - a. A boresighted glide/dive delivery is required.
 - b. Alignment--the leg immediately prior to the bomb release point must be straight, last 2.33 seconds, and must have

a heading within 5° of the heading to the target from the bomb release point.

c. Bomb release range--between 100 and 1,000 meters from the target.

d. Bomb release altitude--between 100 and 2,000 meters.

6. Weapons placement

a. Two type 1 mode 1, two type 2 mode 1, one type 3 mode 4, and one type 3 mode 3 weapons are available for defense placement. Neither of the type 3 weapons may be placed within 3,000 meters of the center of the bridge. A type 5 mode 3 weapon is fixed at (X, Y, Z).

b. One type 5 SAM system is to be placed anywhere on the roads eastward of the six kilometer X axis position.

F. Begin the flight path at an entry point of your choosing along the western boundary and end it along the northern or southern boundary. Anticipate the AAA and SAM placement for bridge defense and plan your flight path accordingly.

G. Locate the AAA and SAM weapons given in the order of battle to best defend the bridge against your opponent's attacking aircraft.

IV. INPUT DATA FORMAT

Three preprocessor programs for POO1 and MICE II are available at the Naval Postgraduate School. These programs create the scenario, flight path, aircraft vulnerability, and weapon location and characteristics data files necessary for the execution of POO1 and MICE II. The preprocessing programs are GRPIP (Graphic POO1/MICE II Input Program), KBPIP (Keyboard POO1/MICE II Input Program) and IBMPIP (IBM POO1/MICE II Input Program). GRPIP is for use on the TEKTRONIX 4010 family of terminals, KBPIP can be used at any keyboard, and IBMPIP runs on IBM 3277 graphics terminals. All of these programs operate in the same fashion, only the manner of data entry is different.

A. PROCEDURE FOR EXECUTING POO1 AND MICE II

1. Log on to VMS
2. ENTER: CP LINK 191 192 R
the message "ENTER READ PASSWORD:" will appear
ENTER:
ENTER: ACC 192 B
 - a. If you have never used GRPIP or IBMPIP, the following file contains the points for drwing the scenario map.
ENTER: COPYFILE MAP DATA B = = A
3.
 - a. To use GRPIP, ENTER: GRPIP
 - b. To use KBPIP, ENTER: KBPIP
 - c. To use IBMPIP, ENTER: IBMPIP
 - d. Execute the selected preprocessor program (see the following sections
4. To execute POO1
 - a. XEDIT file POO1 DATA and insert your JOB card as the first line (see computer center manual MVS-01 pg. 18)
 - b. Then FILE POO1 DATA
 - c. ENTER: SUBMIT POO1 DATA
5. To Execute MICE II
 - a. XEDIT MICE II DATA and insert your JOB card as the first line.
line
 - b. Then FILE MICE DATA
 - c. ENTER: SUBMIT MICE DATA

B. EXECUTING GRPIP (TEKTRONIX Terminals Only)

NOTE: The TEKTRONIX terminal at the computer center does not react to the "RETURN" button. Push "CTRL" and "S" simultaneously for a carriage return.

NOTE: For the terminal at the computer center, the cursor will disappear, but the entry will not be accepted when the key is pressed. On this terminal, press "CTRL" "S" after pressing a key to enter the data.

1. Enter: GRPIP

The following message will appear:

NOTICE YOU ARE LINKED TO 491 AS D FOR TEKTRONIC ROUTINES

This is followed by a delay as the program loads. When loading is complete, the message "EXECUTION BEGINS" will appear, followed by the screen flashing twice.

2. Initialization

The program will request data necessary to initialize the system.

The following message will appear:

GUNS: 0 = DISK FILE; 1 = TERMINAL READ; 2 = PRESET

a. FILE: This reads the file created on a previous run using option 1. Do not select this option if you have never entered gun and missile locations at the terminal.

b. TERMINAL: With this option you enter your own locations.

The gun and missile locations will be saved for later use.

c. PRESET: GRPIP has preset gun and missile sites if you want to use them.

The next message you will see will be:

MILESTONE INPUT: 0 = DISK FILE; 1 = TERMINAL

a. DISK: This reads a previously created file. Do not use this option if you have never run GRPIP, KBPIP, or IBMPPIP.

b. TERMINAL: This allows you to create a new flight path.

When this is done, the following message will be displayed:

ERROR CHECKING: 0 = NO CHECKING; 1 = CHECK FOR ERRORS

a. NO CHECKING: With this option game and flight rule errors will be ignored.

b. CHECK FOR ERRORS: With this option, errors will result in the operator being notified of the cause. When the terminal input option is in effect, the milestone data will be ignored and new data requested. If the disk input option is in effect, the user will see the following prompt if an error occurs during execution:

DO YOU WISH TO FIX THE ERROR: 0 = NO, USE THE POINT 1 = YES

a. 0: The program will ignore the error and use the data.

b. 1: The program will request new data for the current milestone only. Subsequent data will be obtained from the disk. Note that the correction may cause errors in subsequent milestones.

The next message to appear will be:

FLIGHT AND GAME PARAMETERS: 0 = DEFAULT; 1 = USER INPUT

a. DEFAULT: This uses preset game and flight parameters for error checking.

b. USER INPUT: This allows the user to reset error parameters to reflect different game and flight rules. See PARAMETER CHANGING for further information.

3. Gun and Missile Entry

Following initialization, the screen will go blank and the following message will appear:

VILLAGE: 0 = NOT DRAWN; 1 = VILLAGE DRAWN

a. NOT DRAWN: This option results in only the gun, SAM, and target locations being drawn. This saves a great deal of time, particularly on 300 BAUD terminals.

b. VILLAGE DRAWN: This option draws the village, road, and river. When inputting the gun and SAM sites, this option should be chosen to meet the game requirements. Drawing the full map takes about a minute at 1,200 BAUD terminals and about 5 minutes at the slower ones.

If the preset (option 0) or file (option 2) locations are used for the gun and missile sites, they will be drawn immediately. The gun sites are represented by a "+" surrounded by a dotted ring at the engagement radius. The SAM site is a "+" with a small circle around it. The target is a "+" with two small circles around it. User input of the gun sites is as follows:

a. The message ENTER GUN COORDINATES will be underlined.

b. A bell will sound and the map border will flash.

c. A cursor will appear.

d. Position the cursor with the thumbwheels on the right of the terminal.

e. Press any key. The cursor will disappear, a point will appear, and the cursor will re-appear. If the cursor reappears, but not the point, or if you change your mind, press "N" to cancel the input.

f. Press any key except "N" to accept the X and Y values.

g. A bell will ring and the altimeter border will flash.

h. Position the cursor in the altimeter and enter the point as above. Note that the altimeter is marked in 100's of meters.

i. Repeat steps b-h five times. The guns are input in the following order:

(1) Two Type 1 mode 1--1,000m engagement radius

(2) Two Type 2 mode 1--1,400m engagement radius

(3) One Type 3 mode 4--2,500m engagement radius

(4) One Type 3 mode 3--2,500m engagement radius

NOTE: The type 3 guns may not be within 3,000 meters of the target. If you attempt this, the message T00 CLOSE TO TGT will be underlined and a bell will ring several times. The point will be rejected and you will have to enter it again.

NOTE: Entering a weapon (gun or missile) altitude greater than 1,000 meters will abort the program and destroy a previously created GUN LOC file.

When all six gun sites are correctly entered, the message ENTER MISSILE LOCATION will be underlined. Repeat steps b-h.

NOTE: The X coordinate must be greater than 6,000 meters or the error message X COORDINATE < 6,000 will be presented.

4. Milestone Entry

When you have finished entering the gun and missile sites, the bell will ring and the message ENTER MILESTONES will be highlighted in the error section. Enter milestones using steps 3.b-h with the following exceptions:

a. Pressing "A": This signifies that an aiming line is desired.

A dashed line to the target will appear to assist in lining up for the bomb release leg.

b. Pressing "B": This signifies the bomb release milestone.

c. Pressing "R": This resets the problem. The screen will go blank and you will be asked the milestone at which to reset the flight path. This destroys the map and it must be redrawn. This can cause a long wait at the slow terminals.

d. Pressing "S": This stops the program. The point is plotted, but not checked for errors.

The special keys must be pressed when the point is verified, not when it is first selected.

e. The velocity must be entered for each milestone. The horizontal axis of the altimeter is velocity. The tic marks are at 50 meter/sec intervals from 50 to 300 meter/sec. If error checking is used, the minimum speed is 90 meter/sec. Velocity is limited to 260 meter/sec before, and 310 meter/sec after bomb release. Any time a game rule or flight path error is detected, the error will be marked by underlining and ringing the bell. Whenever an error occurs, the X/Y/Z and velocity values are rejected. The Bomb, Reset, and Stop commands take priority over errors while the Aim is executed only at legal milestones. This means that the operator can always stop or reset the program.

f. Repeat steps 1-d until all milestones are entered. See Section E (Output Selection) to finish the program.

C. EXECUTING KBPIP

1. Enter: KBPIP

The terminal will display the file definitions for KBPIP. This is followed by a delay as the program loads. When loading is

complete, the message "EXECUTION BEGINS" will appear, followed by the screen clearing.

2. Initialization

The program will request data necessary to initialize the system.

The following message will appear:

GUNS: 0 = DISK FILE; 1 = TERMINAL READ; 2 = PRESET

a. FILE: This reads the file created on a previous run using option 1. Do not select this option if you have never entered gun and missile locations at the terminal.

b. TERMINAL: With this option you enter your own locations.

The gun and missile locations will be saved for later use.

c. PRESET: GRPIP has preset gun and missile sites if you want to use them.

The next message you will see will be:

MILESTONE INPUT: 0 = DISK FILE; 1 = TERMINAL

a. DISK: This reads a previously created file. Do not use this option if you have never run GRPIP, KBPIP, or IBMPIP.

b. TERMINAL: This allows you to create a new flight path.

When this is done, the following message will be displayed:

ERROR CHECKING: 0 = NO CHECKING; 1 = CHECK FOR ERRORS

a. NO CHECKING: With this option game and flight rule errors will be ignored.

b. CHECK FOR ERRORS: With this option, errors will result in the operator being notified of the cause. When the terminal input option is in effect, the milestone data will be ignored and new data requested.

If the disk input option is in effect, the user will see the following prompt if an error occurs during execution:

DO YOU WISH TO FIX THE ERROR: 0 = NO, USE THE POINT 1 = YES

a. 0: The program will ignore the error and use the data.

b. 1: The program will request new data for the current milestone only. Subsequent data will be obtained from the disk. Note that the correction may cause errors in subsequent milestones.

The next message to appear will be:

FLIGHT AND GAME PARAMETERS: 0 = DEFAULT; 1 = USER INPUT

a. DEFAULT: This uses preset game and flight parameters for error checking.

b. USER INPUT: This allows the user to reset error parameters to reflect different game and flight rules. See PARAMETER CHANGING for further information.

3. Gun and Missile Entry

If preset (option 0) or file (option 2) locations are used, the gun and missile sites will be automatically entered. User input of the gun sites is as follows:

a. The message ENTER GUN LOCATIONS will appear, followed by:

b. ENTER X AND Y COORDINATES.

c. Enter the desired values.

d. The message ENTER ALTITUDE will appear.

e. Enter the desired altitude.

f. Repeat b-e five times. The guns are input in the following order:

- (1) Two Type 1 mode 1-- 1,000m engagement radius
- (2) Two Type 2 mode 1--1,400m engagement radius
- (3) One Type 3 mode 4--2,500m engagement radius
- (4) One Type 3 mode 3--2,500m engagement radius

NOTE: The type 3 guns may not be within 3,000 meters of the target. If you attempt this, the message TOO CLOSE TO TGT will be displayed. The point will be rejected and you will have to enter it again.

NOTE: Entering a weapon (gun or missile) altitude greater than 1,000 meters will abort the program and destroy a previously created GUN LOC file.

When all six gun sites are correctly entered, the message ENTER MISSILE LOCATION will be displayed. Repeat steps b-e.

NOTE: The X coordinate must be greater than 6,000 meters or the error message X COORDINATE < 6,000 will be presented.

4. Milestone Entry

When you have finished entering the gun and missile sites, the message ENTER MILESTONES will be displayed. Enter milestones using the following steps:

- a. The message ENTER X AND Y COORDINATES will be displayed.
- b. Enter the desired coordinates.
- c. The message ENTER ALT., VEL., AND COMMAND: 0 = NONE;

1 = AIM; 2 = BOMB; 3 = RESET; 4 = STOP will then appear.

d. Enter the desired velocity and command. The results of commands are:

(0) NONE: no special action occurs.

(1) AIM: This will provide the distance to the target and the X and Y components for 100 meters along the aiming line.

(2) BOMB: The bomb release rules are checked.

(3) RESET: The program will ask you at what point you want to reset the program. If selected point is before the bomb release point, the release checks will be cancelled.

(4) STOP: The final milestone is accepted and milestone input terminates. If this command is given on the first milestone, the program will stop immediately, otherwise the new milestone file is written and output options are requested before stopping. See Section E.

Any time a game rule error is detected, the error will be displayed. Whenever an error occurs, the X/Y/Z/VEL values are rejected. The Bomb, Reset, and Stop commands take priority over an error, while the Aim command will be accepted only at legal milestones. This means the operator can always get out by executing a Stop.

D. EXECUTING IBMPIP (TEKTRONIX Terminals Only)

1. Enter: IBMPIP

The following message will appear: EXECUTION BEGINS.

This is followed by a delay as the program loads. When loading is complete, the message "EXECUTION BEGINS" will appear, followed by the screen flashing twice.

2. Initialization

The program will request data necessary to initialize the system. The following message will appear:

GUNS: 0 = DISK FILE; 1 = TERMINAL READ; 2 = PRESET

a. FILE: This reads the file created on a previous run using option 1. Do not select this option if you have never entered gun and missile locations at the terminal.

b. TERMINAL: With this option you enter your own locations.

The gun and missile locations will be saved for later use.

c. PRESET: GRPIP has preset gun and missile sites if you want to use them.

The next message you will see will be:

MILESTONE INPUT: 0 = DISK FILE; 1 = TERMINAL

a. DISK: This reads a previously created file. Do not use this option if you have never run GRPIP, KBPIP, or IBMPPIP.

b. TERMINAL: This allows you to create a new flight path.

When this is done, the following message will be displayed:

ERROR CHECKING: 0 = NO CHECKING; 1 = CHECK FOR ERRORS

a. NO CHECKING: With this option game and flight rule errors will be ignored.

b. CHECK FOR ERRORS: With this option, errors will result in the operator being notified of the cause. When the terminal input option is in effect, the milestone data will be ignored and new data requested. If the disk input option is in effect, the user will see the following prompt if an error occurs during execution:

DO YOU WISH TO FIX THE ERROR: 0 = NO, USE THE POINT 1 = YES

a. 0: The program will ignore the error and use the data.

b. 1: The program will request new data for the current milestone only. Subsequent data will be obtained from the disk. Note that the correction may cause errors in subsequent milestones.

The next message to appear will be:

FLIGHT AND GAME PARAMETERS: 0 = DEFAULT; 1 = USER INPUT

a. DEFAULT: This uses preset game and flight parameters for error checking.

b. USER INPUT: This allows the user to reset error parameters to reflect different game and flight rules. See PARAMETER CHANGING for further information.

3. Gun and Missile Entry

Following initialization, the screen will go blank and the following message will appear:

VILLAGE: 0 = NOT DRAWN; 1 = VILLAGE DRAWN

a. NOT DRAWN: This option results in only the gun, SAM, and target locations being drawn. This saves a great deal of time, particularly on 300 BAUD terminals.

b. VILLAGE DRAWN: This option draws the village, road, and river. When inputting the gun and SAM sites, this option should be chosen to meet the game requirements. Drawing the full map takes about a minute at 1,200 BAUD terminals and about 5 minutes at the slower ones.

If the preset (option 0) or file (option 2) locations are used for the gun and missile sites, they will be drawn immediately. The gun sites are represented by a "+" surrounded by a dotted ring at the engagement radius. The SAM site is a "+" with a small circle around it. The target is a "+" with two small circles around it. User input of the gun sites are as follows:

a. The message ENTER GUN COORDINATES will be displayed. Press the "CLEAR" key on the upper left part of the keyboard.

b. The map border will flash and a cursor will appear.

c. Position the cursor with the joystick on the right of the terminal.

d. Press any light grey key. The cursor will disappear, a point will appear, and the cursor will re-appear.

e. Press any PF key to accept the X and Y values, press a grey key to reject it. Repeat d. and e. if the point is rejected.

f. The altimeter border will flash.

g. Position the cursor in the altimeter and enter the point as above. Note that the altimeter is marked in 100's of meters.

h. Repeat steps b-h five times. The guns are input in the following order:

(1) Two Type 1 mode 1--1,000m engagement radius

(2) Two Type 2 mode 1--1,400m engagement radius

(3) One Type 3 mode 4--2,500m engagement radius

(4) One Type 3 mode 3--2,500m engagement radius

NOTE: The type 3 guns may not be within 3,000 meters of the target. If you attempt this, the message T00 CLOSE TO TGT will be underlined and a bell will ring several times. The point will be rejected and you will have to enter it again.

NOTE: Entering a weapon (gun or missile) altitude greater than 1,000 meters will abort the program and destroy a previously created GUN LOC file.

When all six gun sites are correctly entered, the message ENTER MISSILE LOCATION will appear. Press the "CLEAR" key and repeat steps b-g.

NOTE: The X coordinate must be greater than 6,000 meters or the error message X COORDINATE < 6,000 will be presented.

4. Milestone Entry

When you have finished entering the gun and missile sites, the bell will ring and the message ENTER MILESTONES will appear on the IBM terminal. Press "CLEAR." Enter milestones using steps 3.b-h with the following exceptions:

- a. Pressing "PF1": This signifies that an aiming line is desired. A dashed line to the target will appear to assist in lining up for the bomb release leg.
- b. Pressing "PF2": This signifies the bomb release milestone.
- c. Pressing "PF3": This resets the problem. The screen will go blank and you will be asked the milestone at which to reset the flight path. This destroys the map and it must be redrawn. This can cause a long wait at the slow terminals.
- d. Pressing "PF4": This stops the program. The point is plotted, but not checked for errors.

The special keys must be pressed after the dot has appeared. Press PF keys 5-12 to accept the point without any special option selected.

e. The velocity must be entered for each milestone. The horizontal axis of the altimeter is velocity. The tic marks are at 50 meter/sec intervals from 50 to 300 meter/sec. If error checking is used, the minimum speed is 90 meter/sec. Velocity is limited to 260 meter/sec before, and 310 meter/sec after bomb release. Any time a game rule or flight path error is detected, the error will be marked by a thin rectangle flashing on the right side of the graphics screen. The error message will appear on the IBM screen. Press "CLEAR" and resume entering

milestones. Whenever an error occurs, the X/Y/Z and velocity values are rejected. The Bomb, Reset, and Stop commands take priority over errors while the Aim is executed only at legal milestones. This means that the operator can always stop or reset the program.

E. OUTPUT SELECTION (ALL VERSIONS)

When new milestones or weapons have been entered, the program writes them to disk file PTS LOC for later use. If a new flight path was created, the program then displays:

ARE YOU FINISHED WITH THIS FLIGHT PATH: 0 = NO; 1 = YES

a. NO: This marks the data with an end of data command. When read by GRPIP, KBPIP, or IBMPIP, no further points can be added.

b. YES: This marks the file with a continuation command. When GRPIP, IBMPIP, or KBPIP finish reading the file on a later run, they will ask the operator to continue entering milestones.

The computer then prints:

OUTPUT FILE: 0 = NO OUTPUT; 1 = POO1 FILE ONLY;

2 = MICE FILE ONLY; 3 = POO1 & MICE FILES

The POO1 file is located on your disk as POO1 DATA and the MICE file is called MICE DATA.

F. HARD COPY

There are three methods of obtaining a hard copy of the flight path and weapon sites. The first uses the TEKTRONIX plotter, the second uses the VERSATEC plotter, and the third uses the TEKTRONIX hard copy device.

To use the TEKTRONIX plotter, it is necessary to connect the TEKTRONIX terminal to the plotter and then to the modem. Set the corners and then

leave the plotter on LOAD. Before the final milestone is entered, press "R" for the reset option. After the request for starting point is printed, take the plotter out of LOAD. The plotter will plot the milestones, weapon sites, and, if requested, the village. Finish inputting the last milestone. The plotter will plot the final milestones. Before the output selection is started, set the plotter back to LOAD and finish running the program.

To use the VERSATEC plotter, two methods are available. First, a map of the scenario only can be made. The other option is to get a complete plot of the scenario, milestones, and weapon sites.

To get only the map, perform the following steps:

- a. ENTER: COPY VERSA PLOT B = = A.
- b. XEDIT VERSA PLOT and put your JOB card in as the first entry.
- c. FILE VERSA PLOT.
- d. ENTER: SUBMIT VERSA PLOT.

To get a complete drawing, perform the following:

- a. ENTER: COPY HDR DATA B PTS LOC A PLOT MAP A.
- b. XEDIT PLOT MAP and put your JOB card in as the first entry and /* as the final one.
- c. FILE PLOT MAP.
- d. ENTER: SUBMIT PLOT MAP.

The VERSATEC maps can be picked up at the computer center printer room.

The TEKTRONIX hard copy device is very simple to use. Turn it on and allow it to warm up, about 15 minutes. This warm-up can be done while you are running GRPIP or IBMPIP. When you want a copy, press the

button marked COPY. You will see a vertical line sweep across the terminal screen. Soon after this, the hard copy device will provide your copy. Several copies may be necessary to properly adjust the intensity. On the 622 (IBMPIP) terminal, this intensity can also be adjusted using the knob on the bottom right part of the terminal marked HARD COPY INTENSITY.

G. CHANGING PARAMETERS

When this option is selected, the following message will appear:

```
SELECT PARAMETER TO BE CHANGED:  1 = NO MORE CHANGES
2 = MAX LIFT COEFF; 3 = WING LOADING; 4 = STALL SPEED
5 = MAX G FORCE; 6 = MIN ALT; 7= MAX ALT (< 2,200 MTR)
8 = DISTANCE TO TARGET BEFORE POPUP ALLOWED
9 = MAX APPROACH ALT; 10 = MAX T/W RATIO; 11 = DRAG COEFF. W/O LIFT
12 = LIFT DRAG CONSTANT; 13 = MAX SPD WITH BOMB
14 = MAX SPD W/O BOMB; 15 = TARGET COORDINATES
16 READ PARAMETER FILE; 17 = LIST PARAMETERS
```

Enter the desired value. If a number between 2 and 5 is selected, another prompt will appear. Enter the desired value. If 16 is chosen, disk file PAR SAV will be read; do not choose this option if you have never changed parameters. A 17 will list the current values of the parameters. To exit, select a 1. This will write the current parameter values to PAR SAV and return to the main program.

APPENDIX B

GRPIP PROGRAM LISTING

```

COMMON /LOC/ X1,Y1,Z1,V1,LTR,LTR2
COMMON /MN/ IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
COMMON /PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
COMMON /PAR1/XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
COMMON /PAR2/T(200),XGUN(7),YGUN(7),ZGUN(7),XSAM,YSAM,ZSAM,GR(7)
COMMON /OPT/IGUN,IPNCH,IEXT,ISAM,IMP,KER
MBR=0
MNUM=0
T(1)=0.
CALL BEGIN
CALL SCENE(0,0)
REWIND 10
IF (IGUN .NE. 1) GOTO 250

      IGUN=1==> READ GUN SITES FROM TERMINAL

      CALL ERRMK(16)
      DO 212 I=1,6
211      IERR=0
          CALL XYZIN
          XGUN(I)=X1
          YGUN(I)=Y1
          ZGUN(I)=Z1
          IF (Z1.GE.1000.) STOP
          IF (I.GE. 5) CALL GUNCHK(X1,Y1,IERR)
          IF (IERR.EQ. 0) GOTO 212
          CALL ERRMK(IERR)
          GOTO 211
212      WRITE (10,501) X1,Y1,Z1
220      CALL ERRMK(14)
          CALL XYZIN
          XSAM=X1
          YSAM=Y1
          ZSAM=Z1
          IF (Z1.GE.1000.) STOP
          CALL SAMCHK(X1,IERR)
          IF (IERR.NE. 0) GOTO 220
          WRITE(10,501) X1,Y1,Z1

      IGUN=2==> READ GUN SITES FROM DATA FILE

250      IF (IGUN .NE. 0) GO TO 260
          DO 252 I=1,6
252      READ (10,501) XGUN(I),YGUN(I),ZGUN(I)
          READ(10,501) XSAM,YSAM,ZSAM

      DRAW GUN SITES AND ENGAGEMENT CIRCLES

260      DO 265 I=1,7
265      CALL GUNLOC(XGUN(I),YGUN(I),GR(I))
          CALL GUNLOC(XSAM,YSAM,150.)

```



```
C
C      THIS SECTION ACCEPTS THE FLIGHT MILESTONES
      FTFAC=3.28084
      DGFAC=57.29578
      CALL ERRMK(15)
      REWIND 11
300  MNUM=MNUM+1
C
C      READ MILESTONES FROM THE DISK
302  IF (IMP.NE.0) GOTO 305
      READ(11,625) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),LTR
      IF (X(MNUM).LE.0.1) GOTO 302
      IF (LTR.EQ.1) IMP=1
      LTR2=LTR
      GOTO 308
C
C      READ MILESTONES FROM THE TERMINAL
305  CALL XYZIN
      X(MNUM)=X1
      Y(MNUM)=Y1
      Z(MNUM)=Z1
      VEL(MNUM)=V1
C
C      COMPUTE FLIGHT PARAMETERS AND CHECK FOR ERRORS OR USER COMMANDS
308  IF(MNUM.EQ.1.AND.(LTR.EQ.83.OR.LTR2.EQ.83)) STOP
      IF(MNUM.EQ.1) GOTO 300
      IERR=0
      CALL VALSET(IERR)
      IF(((LTR.EQ.82).OR.(LTR2.EQ.82)).AND.(IMP.NE.0)) GOTO 350
      IF((LTR.EQ.66).OR.(LTR2.EQ.66)) MBR=MNUM
      IF((LTR.EQ.83).OR.(LTR2.EQ.83)) GOTO 370
      IF(IERR.EQ.0) CALL ERRCHK(IERR)
      IF((IERR.KER.EQ.0).AND.(IERR.NE.12)) GOTO 310
      CALL ERRMK(IERR)
      IF(IMP.NE.0) GO TO 305
      CALL ERRMK(20)
      READ(5,*) ICOR
      IF(ICOR.EQ.1) GO TO 305
C
C      MILESTONE ACCEPTED, RESTART THE SEQUENCE
310  CALL PTHPLT
      IF(LTR.EQ.65.OR.LTR2.EQ.65) CALL AIMPT(MNUM)
      GOTO 300
C
C      THIS SECTION RESETS THE DATA
350  CALL SCENE(1,ICT)
      REWIND 19
      IF(MBR.GT. ICT) MBR=0
C
C      COMPLETE RESTART
      IF( ICT.GT. 1) GOTO 352
      MNUM=0
      GOTO 260
C
C      DRAW RETAINED MILESTONES
352  IF ((ICT.GE. MNUM).AND.(MNUM.GE. 3)) ICT=MNUM-1
      DO 355 MNUM=2,ICT
355  CALL PTHPLT
      MNUM=MNUM-1
      GOTO 260
```


CCCCCCCC

370

CCC

371

372

CCCC

—

IGUN=1

WRITE (11,

DO 375 I=

LTR=0
IF (I

$$\frac{IF}{IF} \left\{ \frac{1}{T} \right\} \cdot$$

WRITE } 11.

375

WRITE (11,6 27)

BLANK

IF (1GUN
DO 376

DO NOT
WRITE

376

CCC

377

$$RA(I) = RA$$
$$\text{HDG}(\mathbf{I}) = \mathbf{H}$$

380

FORMAT { 217 }
FORMAT { 3 E 10 }

FORMAT } 3 F 10
FORMAT } 4 F 10

FORMAT { ' 999

FORMAT(F10.

STOP
END

END
CCCCC

C
C

CCCCCCCCCCCC

CCCCCCCCCCCCCCCC
SUBROUTINE

SUBROUTINE
COMMON /PAR/

COMMON / PAR 1

COMMON / PAR 2

COMMON/PAR3

DIMENSION A
CFF-0 22

GEF=9.82


```

DX=X(MNUM)-X(MNUM-1)
DY=Y(MNUM)-Y(MNUM-1)
DZ=Z(MNUM)-Z(MNUM-1)
IF ((DX.NE. 0.) .OR. (DY.NE. 0.)) GOTO 5
IERR=12
RETURN
5 VAVGL=VAVG
DISTL=DIST

```

LIMIT THE SPEED

```

DIST=SQRT(DX**2 + DY**2 + DZ**2)
IF ((VEL(MNUM).GT. VMAX1).AND.(MBR.EQ. 0)) VEL(MNUM)=VMAX1
IF ((VEL(MNUM).GT. VMAX2).AND.(MBR.GE. 1)) VEL(MNUM)=VMAX2
VAVG=(VEL(MNUM)+VEL(MNUM-1))/2.
DT=DIST/VAVG
T(MNUM)=T(MNUM-1)+DT

```

COMPUTE AVERAGE VELOCITY COMPONENTS

```

AXD(MNUM)=VAVG*DX/DIST
AYD(MNUM)=VAVG*DY/DIST
AZD(MNUM)=VAVG*DZ/DIST
IF (MNUM.GE. 3) GOTO 20

```

COMPUTE THE PARAMETERS FOR THE INITIAL LEG OF THE FLIGHT PATH

```

CA(1)=ATAN2(DZ,SQRT(DX**2 + DY**2))
IF ((DX.EQ. 0.) .OR. (DY.EQ. 0.)) GOTO 10
HDG(1)=ATAN2(DY,DX)
GOTO 11
10 IF ((DX.EQ. 0.) .AND. (DY.GT. 0.)) HDG(1)=1.57079
IF ((DX.EQ. 0.) .AND. (DY.LT. 0.)) HDG(1)=-1.57079
IF ((DY.EQ. 0.) .AND. (DX.LT. 0.)) HDG(1)=3.14159
IF ((DY.EQ. 0.) .AND. (DX.GT. 0.)) HDG(1)=0.
11 RA(1)=0.
XDOT(1)=VEL(1)*DX/DIST
YDOT(1)=VEL(1)*DY/DIST
ZDOT(1)=VEL(1)*DZ/DIST
RETURN
20 DELT=(T(MNUM)-T(MNUM-2))/2.

```

COMPUTE THE ACCELERATION COMPONENTS

```

XDD(MNUM-1)=(AXD(MNUM)-AXD(MNUM-1))/DELT
YDD(MNUM-1)=(AYD(MNUM)-AYD(MNUM-1))/DELT
ZDD(MNUM-1)=(AZD(MNUM)-AZD(MNUM-1))/DELT

```

WEIGHT AVERAGE OF THE VELOCITY COMPONENTS

```

HDAVG=DISTL/(DIST+DISTL)
TDX=(AXD(MNUM)/VAVG-AXD(MNUM-1)/VAVGL)*HDAVG+AXD(MNUM-1)/VAVGL
TDY=(AYD(MNUM)/VAVG-AYD(MNUM-1)/VAVGL)*HDAVG+AYD(MNUM-1)/VAVGL
TDZ=(AZD(MNUM)/VAVG-AZD(MNUM-1)/VAVGL)*HDAVG+AZD(MNUM-1)/VAVGL

```

MAKE THE WEIGHTED VALUES A UNIT VECTOR AND COMPUTE THE MILESTONE VELOCITY COMPONENTS

```

UNIT=SQRT(TDX*TDX + TDY*TDY + TDZ*TDZ)
IF (ABS(UNIT).GE. 0.01) GO TO 25
IERR=12
RETURN
25 TDX=TDX/UNIT
TDY=TDY/UNIT
TDZ=TDZ/UNIT
XDOT(MNUM-1)=VEL(MNUM-1)*TDX
YDOT(MNUM-1)=VEL(MNUM-1)*TDY
ZDOT(MNUM-1)=VEL(MNUM-1)*TDZ
IF (SQRT(TDX*TDX+TDY*TDY).GE. 0.01) GO TO 30
CA(MNUM-1)=1.5533
GOTO 33
30 CA(MNUM-1)=ATAN2(TDZ,SQRT(TDX*TDX+TDY*TDY))

```


CCC

45

ccc

2

[illegible]

CCC

C

C

C

[illegible]


```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

SUBROUTINE PTHPLT
COMMON/PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
COMMON/PAR1/XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
COMMON/MN/IBAUD,MINY,MAXY,MINX,MAXX

```

```

IDENTIFY THE MAP, DRAW AND LABEL THE FLIGHT LEG

```

```

CALL WIN(MINX,MAXX,18000.,12000.,0)
CALL MOVEA(X(MNUM-1),Y(MNUM-1))
CALL DRAWA(X(MNUM),Y(MNUM))
CALL ANMCDE
IF (MNUM.GT.99) WRITE(6,900) MNUM
IF ((MNUM.GT.9).AND.(MNUM.LT.100)) WRITE(6,901) MNUM
IF (MNUM.LE.9) WRITE(6,902) MNUM

```

```

SAVE THE POINT IN 'TEMP DATA'

```

```

625 WRITE(19,625) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),MBR
900 FORMAT(4F10.0,I3)
901 FORMAT('+',I3)
902 FORMAT('+',I2)
RETURN
END

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

SUBROUTINE GUNLOC(GX,GY,RAD)
COMMON/MN/IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1

```

```

IDENTIFY THE MAP AND PLACE A '+' AT THE LOCATION

```

```

CALL WIN(MINX,MAXX,18000.,12000.,0)
CALL MOVEA(GX-75.,GY)
CALL DRAWA(GX+75.,GY)
CALL MOVEA(GX,GY+75.)
CALL DRAWA(GX,GY-75.)

```

```

DETERMINE NUMBER OF STEPS TO BE USED

```

```

ISTEP=3
IF (RAD.LT.1600.) ISTEP=6

```

```

DRAW A CIRCLE AROUND THE SITE

```

```

DO 10 I=3,180,ISTEP
  ANGLE=I*0.034907
  DX=RAD*COS(ANGLE)+GX
  DY=RAD*SIN(ANGLE)+GY
  CALL MOVEA(DX+10.,DY)
  CALL DRAWA(DX-10.,DY)
10 RETURN
END

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

SUBROUTINE SPOT(X,Y,L1,L2)

```

```

READ THE CURSOR

```

```

100 CALL VCURSR(L1,X,Y)

```

```

MARK THE SPOT

```

```

CALL POINTA(X,Y)

```


133


```

CALL MOVREL(0,-20)
CALL ANMODE
WRITE(6,503)
CALL MOVEA(8000.,0.)
CALL MOVREL(0,-40)
CALL ANMODE
WRITE(6,504)

```

DRAW THE ALTIMETER

```

CALL WIN(MIN1,MAX1,350.,2200.,1)
DO 220 I=2,20,2
  CALL MOVEA(0.,I*100.)
  CALL DRAWA(350.,I*100.)
  CALL ANMODE
  WRITE(6,502) I
DO 225 I=1,6
  CALL MOVEA(I*50.,100.)
  CALL DRAWA(I*50.,0.)

```

MARK RESET POINT VELOCITY AND ALTITUDE AS OPERATOR AID

```

IF (IRQ.NE.0 .AND. ICT .GT.1) CALL POINTA(VEL(ICT),Z(ICT))
CALL MOVEA(0.,0.)
CALL MOVREL(0,-15)
CALL ANMODE
WRITE(6,505)
CALL MOVEA(0.,0.)
CALL MOVREL(0,-30)
CALL ANMODE
WRITE(6,506)
DO 499 IT=1,20
  I=2
  IF (IT .LE. 6) I=1
  IF (IT .GE. 13) I=3
  CALL MOVABS(MKRX(I),MKRY(IT))
  CALL ANMODE
  GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20),IT
1 WRITE(6,601)
  GOTO 499
2 WRITE(6,602)
  GOTO 499
3 WRITE(6,603)
  GOTO 499
4 WRITE(6,604)
  GOTO 499
5 WRITE(6,605)
  GOTO 499
6 WRITE(6,606)
  GOTO 499
7 WRITE(6,607)
  GOTO 499
8 WRITE(6,608)
  GOTO 499
9 WRITE(6,609)
  GOTO 499
10 WRITE(6,610)
  GOTO 499
11 WRITE(6,611)
  GOTO 499
12 WRITE(6,612)
  GOTO 499
13 WRITE(6,613)
  GOTO 499
14 WRITE(6,614)
  GOTO 499
15 WRITE(6,615)
  GOTO 499
16 WRITE(6,616)
  GOTO 499
17 WRITE(6,617)
  GOTO 499

```


[illegible]

C

52*9.853,6*10.51,2*9.639,6*11.15,2*12.64,6*14.78,2*9.639,6*11.15,2*13.47/
612.54,6*13.47/

VULNERABLE AREA TABLE VS TYPE 5 WEAPONS

DATA VAT5/8*55.37,8*43.22,8*47.10,8*62.53,8*47.10,8*43.22,8*47.10,
18*62.53,8*47.10,8*5.761,8*27.45,8*33.07,8*27.45,8*5.761,8*27.45,8*
233.07,8*27.45,8*5.761,8*47.10,8*62.53,8*47.10,8*43.22,8*47.10,8*62.53,
3.53,8*47.10,8*55.37/

RADAR CROSS SECTION TABLE

DATA RCSTAB/19*1000.,19*100.,19*10.,19*10.,19*10.,19*100.,
+19*1000.,867*0.0/

CALL ERRSET SUPPRESSES ANY POSSIBLE UNDERFLOW PROBLEMS THAT MAY
RESULT FROM MANIPULATION OF SCENARIO PARAMETERS.

CALL ERRSET (208,50,-1,1,1)
JAM=0

CARD 2 TIME INCREMENT CALCULATION

TINC = T(MNUM)/1000. + 0.0008

CARD 6 TIME INCREMENT CALCULATIONS

TINKI = 0

DO 10 I=1,9
TINK(I) = TINKI+T(MNUM)/10
TINKI = TINK(I)
10 CONTINUE

/////*****PUNCH PROGRAM*****/////

OPTION TO PUNCH THE P001 CARD DECK OR MICE-II CARD DECK

IF (IPNCH.EQ.0) RETURN
IF (IPNCH.EQ.2) GO TO 155

COMMENCE PUNCHED OUTPUT OF THE P001 CARD DECK.

THE JCL CARDS.

WRITE (18,79)
WRITE (18,80)
WRITE (18,81)
WRITE (18,82)
WRITE (18,83)
WRITE (18,84)
WRITE (18,85)
WRITE (18,86)
WRITE (18,87)

LEADING BLANK DATA CARD SIGNIFIES RADAR MASKING ANGLE OF ZERO.

THE OUTPUT TITLE CARD.

WRITE (18,90)

CARD 2

WRITE (18,88)
WRITE (18,91) T(MNUM),TINC

THE 2A CARDS (MILESTONES).


```

C
C
C 4000 DO 17 I=1,MNUM
      WRITE (18,92) T(I),X(I),Y(I),Z(I),XDOT(I),YDOT(I),ZDOT(I),HDG(I)
      1,CA(I),RA(I)
      17 CONTINUE
C
C
C      WRITE (18,94)
C
C CARD 3 (GUN EMPLACEMENT CARD) .
C
C      WRITE (18,95) XGUN(1),YGUN(1),ZGUN(1)
C
C CARD 4 (GUN TYPE) .
C
C      WRITE (18,93)
C      WRITE (18,96)
C
C CARD 5
C
C      WRITE (18,89)
C      WRITE (18,97)
C
C CARD 6
C
C      WRITE (18,398)
C      WRITE (18,98) (TINK(I),I=1,9)
C
C CARD 7 (VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS)
C
C      WRITE (18,399)
C      WRITE (18,99)
C      WRITE (18,100) (VAT1N2(I),I=1,208)
C
C
C CARD 12 (EXECUTE RUN).
C EXTENDED OUTPUT OPTION
C
C      WRITE (18,3101)
C      IF (IEXT.NE.1) WRITE (18,102)
C      IF (IEXT.EQ.1) WRITE (18,101)
C
C THE REMAINDER OF THE CARDS INTRODUCE NEW GUN LOCATIONS, GUN TYPES
C AND VULNERABLE AREA TABLES TO BE EXECUTED BY THE PROGRAM.
C
C      WRITE (18,94)
C      WRITE (18,95) XGUN(2),YGUN(2),ZGUN(2)
C
C EXTENDED OUTPUT OPTION
C
C      WRITE (18,3101)
C      IF (IEXT.NE.1) WRITE (18,102)
C      IF (IEXT.EQ.1) WRITE (18,101)
C      WRITE (18,94)
C      WRITE (18,95) XGUN(3),YGUN(3),ZGUN(3)
C      WRITE (18,93)
C      WRITE (18,103)
C
C EXTENDED OUTPUT OPTION
C
C      WRITE (18,3101)
C      IF (IEXT.NE.1) WRITE (18,102)
C      IF (IEXT.EQ.1) WRITE (18,101)
C      WRITE (18,94)
C      WRITE (18,95) XGUN(4),YGUN(4),ZGUN(4)
C
C EXTENDED OUTPUT OPTION
C
C      WRITE (18,3101)

```



```

IF (TEXT.NE.1) WRITE (18,102)
IF (TEXT.EQ.1) WRITE (18,101)
WRITE (18,94)
WRITE (18,95) XGUN(5), YGUN(5), ZGUN(5)
WRITE (18,93)
WRITE (18,104)

```

CARD 7 (VULNERABLE AREA TABLE VS TYPE 3 WEAPONS)

```

WRITE (18,399)
WRITE (18,149)
WRITE (18,100) (VAT3(I), I=1,208)

```

EXTENDED OUTPUT OPTION

```

WRITE (18,3101)
IF (TEXT.NE.1) WRITE (18,102)
IF (TEXT.EQ.1) WRITE (18,101)
WRITE (18,94)
WRITE (18,95) XGUN(6), YGUN(6), ZGUN(6)
WRITE (18,93)
WRITE (18,107)

```

EXTENDED OUTPUT OPTION

```

WRITE (18,3101)
IF (TEXT.NE.1) WRITE (18,102)
IF (TEXT.EQ.1) WRITE (18,101)
WRITE (18,94)
WRITE (18,95) XGUN(7), YGUN(7), ZGUN(7)
WRITE (18,93)
WRITE (18,109)

```

CARD 7 (VULNERABLE AREA TABLE VS TYPE 5 WEAPON)

```

WRITE (18,399)
WRITE (18,150)
WRITE (18,100) (VAT5(I), I=1,208)

```

EXTENDED OUTPUT OPTION

```

WRITE (18,3101)
IF (TEXT.NE.1) WRITE (18,102)
IF (TEXT.EQ.1) WRITE (18,101)
WRITE (18,110)
22 IF (IPNCH.EQ.1) RETURN
C  FORMAT STATEMENTS
40 FORMAT (3F10.0)
41 FORMAT (3F10.0)
42 FORMAT (F10.0, I2, 8I1, F10.0)
60 FORMAT (1X, 10(F7.1, 1X))
61 FORMAT (1X, ' ')
68 FORMAT (1X, 8F8.3)
79 FORMAT ('// EXEC PGM=P1AD')
80 FORMAT ('//STEPLIB DD DISP=SHR,DSN=MSS.F0559.PIPSAV')
81 FORMAT ('//GO.FT07F001 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),')
82 FORMAT ('//DCB=(RECFM=VBS,LRECL=404,BLKSIZE=3236)')
83 FORMAT ('//GO.FT09F001 DD DUMMY')
84 FORMAT ('//GO.FT06F001 DD SYSOOT=A')
85 FORMAT ('//GO.FT05F001 DD *')
86 FORMAT ('1,1,0,0,0,0')
87 FORMAT ('01')
88 FORMAT ('02')
89 FORMAT ('05')
90 FORMAT (' AIRCRAFT COMBAT SURVIVABILITY SCENARIO')
91 FORMAT ('0,12,0,'F7.2','100000.','F7.4','/')
92 FORMAT (10(F7.1, 1X))
93 FORMAT ('04')
94 FORMAT ('03')
95 FORMAT (3(1X,F7.0),' 0.0,360.0')

```



```

96 FORMAT ('0,1,1,1,1,1,0.0,50.')
97 FORMAT ('1,1,1.0')
398 FORMAT ('06')
98 FORMAT ('1,9',9(1X,F7.3))
399 FORMAT ('07')
99 FORMAT (' VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS')
100 FORMAT ('8F8.3')
3101 FORMAT ('12')
101 FORMAT ('1,1,1,1,1,1,1')
102 FORMAT ('0,0,0,0,1,1,1')
103 FORMAT ('0,2,1,1,1,0.0,50.0')
104 FORMAT ('0,3,4,1,1,0.0,50.0')
105 FORMAT ('14',I3,5,0,1.0,7X,F6.1,4X,'1.0E-06',6X,'1',11X,
1 '1.0',6X,I5,F10.2,/'GEND',/,3X,I9,4X,'99.0',180.0,11X,
2 '1000.0')
106 FORMAT ('8F10.3,/,8F10.3,/,3F10.3')
107 FORMAT ('0,3,3,4,1,1,0.0,50.0')
108 FORMAT ('13',I3,I5,'0.35')
109 FORMAT ('0,5,3,2,1,1,0.0,50.0')
110 FORMAT ('/')
149 FORMAT (' VULNERABLE AREA TABLE VS TYPE 3 WEAPONS')
150 FORMAT (' VULNERABLE AREA TABLE VS TYPE 5 WEAPON')
155 CONTINUE

C COMMENCE PUNCHED OUTPUT OF THE MICE-II CARD DECK
C
C CHECK FOR VALID MISSILE DESIGNATION NUMBER FOR MICE II
C IF (ISAM.LE.7.AND.ISAM.GE.1) GO TO 156
C RETURN

C THE JCL CARDS
C
156 WRITE (17,200)
WRITE (17,201)
WRITE (17,202)
WRITE (17,203)

C THE OUTPUT TITLE CARDS
C
WRITE (17,204)
WRITE (17,205)

C THE PROBLEM RUN INPUT CARDS, CHECKING FOR TYPE OF MISSILE
C TO INPUT TO MICE II
C
WRITE (17,206)
IF (ISAM.EQ.1) WRITE (17,250) UP 10
IF (ISAM.EQ.2) WRITE (17,251)
IF (ISAM.EQ.3) WRITE (17,252)
IF (ISAM.EQ.4) WRITE (17,253)
IF (ISAM.EQ.5) WRITE (17,254)
IF (ISAM.EQ.6) WRITE (17,255)
IF (ISAM.EQ.7) WRITE (17,256)

C THE PSSK CALCULATION CARDS
C
WRITE (17,208)
WRITE (17,209)

C THE RADAR CROSS SECTION TABLE
C
WRITE (17,210)
WRITE (17,211)
WRITE (17,212)
WRITE (17,213)
WRITE (17,214) (RCSTAB(I),I=1,133)

C THE SIMULATION TIME PARAMETERS
C
WRITE (17,215)
WRITE (17,216)

```


C

C

C

C

C

C

2

C

2

C

C

CCCCC
CCCCC

ELFIN

CC CCC

SUBROUTINE ELFIN (LAST)


```

      END
      SUBROUTINE SAMCHK(X,IERR)
      IERR=0
      IF(X.GE.6000.) RETURN
      CALL ERRMK(13)
      IERR=1
      RETURN
      END
      SUBROUTINE AIMPT(I)

```



```

14      GO TO 30
      WRITE(6,214)
      READ(5,*) VMAX2
      GO TO 30
15      WRITE(6,215)
      READ(5,*) TARGX,TARGY
      GO TO 30

C      READ NEW PARAMETER VALUES FROM THE DISK
C
16      REWIND 16
      READ (16,300) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      READ (16,300) POPMIN,APPMAX,TMAX,CD0,CDK,VMAX1
      READ (16,300) VMAX2,TARGX,TARGY
      GO TO 30

C      WRITE PARAMETER VALUES TO THE TERMINAL
C
17      WRITE(6,250)
      WRITE(6,300) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE(6,255)
      WRITE(6,300) POPMIN,APPMAX,TMAX,CD0,CDK,VMAX1
      WRITE(6,260)
      WRITE(6,300) VMAX2,TARGX,TARGY
      GO TO 30

C      SAVE PARAMETER VALUES ON DISK AND RETURN TO CALLING ROUTINE
C
200     REWIND 16
      WRITE(16,300) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE(16,300) POPMIN,APPMAX,TMAX,CD0,CDK,VMAX1
      WRITE(16,300) VMAX2,TARGX,TARGY
      RETURN

100     FORMAT(' ')
105     FORMAT(' SELECT PARAMETER TO BE CHANGED: 1=NO MORE CHANGES')
110     FORMAT(' 2=MAX LIFT COEFF; 3=WING LOADING; 4=STALL SPEED')
115     FORMAT(' 5=MAX G FORCE; 6=MIN ALT; 7=MAX ALT(<2200 MTR)')
120     FORMAT(' 8=DISTANCE TO TARGET BEFORE POPUP ALLOWED')
125     FORMAT(' 9=MAX APPROACH ALT; 10=MAX T/W RATIO; 11=DRAG COEF. W/O
$ LIFT')
130     FORMAT(' 12=LIFT DRAG CONSTANT; 13=MAX SPD WITH BOMB')
135     FORMAT(' 14=MAX SPD W/O BOMB; 15=TARGET COORDINATES ')
140     FORMAT(' 16=READ PARAMETER FILE; 17=LIST PARAMETERS')
202     FORMAT(' ENTER MAX LIFT COEFFICIENT')
203     FORMAT(' ENTER WING LOADING')
204     FORMAT(' ENTER STALL SPEED')
205     FORMAT(' ENTER MAX G FORCE ALLOWED')
206     FORMAT(' ENTER MINIMUM ALTITUDE')
207     FORMAT(' ENTER MAXIMUM ALTITUDE (LESS THAN 2000 METERS)')
208     FORMAT(' ENTER MINIMUM DISTANCE TO TARGET BEFORE POP-UP ALLOWED')
209     FORMAT(' ENTER MAXIMUM APPROACH ALTITUDE')
210     FORMAT(' ENTER MAX THRUST TO WEIGHT ALLOWED')
211     FORMAT(' ENTER DRAG COEFFICIENT FOR ZERO LIFT')
212     FORMAT(' ENTER LIFT DRAG CONSTANT')
213     FORMAT(' ENTER MAX SPEED CARRYING A BOMB')
214     FORMAT(' ENTER MAX SPEED AFTER BOMB IS RELEASED')
215     FORMAT(' ENTER TARGET X AND Y COORDINATES')
250     FORMAT(' MAX CL WING LD STALL SPD MAX G MIN.HT. MAX.HT.')
255     FORMAT(' POP-UP DIST APPR.HT. MAX THRUST CD0 DRAG CONSTANT
*MAX SPD WITH BOMB')
260     FORMAT(' MAX SPD W/O BOMB TARGET X COORD. TARGET Y COORD.')
300     FORMAT(6F12.4)
      END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      SUBROUTINE ERRCHK
C      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ERRCHK(IERR)
      COMMON/PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
      COMMON/PAR1/XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
      COMMON/PAR2/T(200)

```



```

COMMON/TAR/TARGX,TARGY
COMMON/PAR3/TMD,ACLIFT,CLMAX,WL,TMAX,CDO,CDK
COMMON/PAR4/APPMAX,HTMIN,HTMAX,SPDMIN,GMAX,POPMIN
DATA TMSAV/-1./
DX=TARGX-X(MNUM)
DY=TARGY-Y(MNUM)
DIST=SQRT(DX**2 + DY**2)

```

```

C
C
C INITIALIZE POPALT AND TMSAV

```

```

15 IF ((MNUM.NE.2) .AND. (TMSAV.NE.-1.)) GO TO 15
    POPALT=0.
    TMSAV=TMAX
IF (MBR.EQ.0 .AND. TMAX.NE.TMSAV) TMAX=TMSAV
CALL ERRMK(22)
IF (MBR.GE.1) GOTO 30
IF ((DIST.LT. POPMIN).OR. (Z(MNUM).LT. APPMAX)) GOTO 30
    IERR=4
    RETURN
30 IF (DIST.LE. POPMIN .AND. Z(MNUM).GT. POPALT) POPALT=Z(MNUM)
35 IF (Z(MNUM).GE. HTMIN) GOTO 40
    IERR=3
    RETURN
40 IF (Z(MNUM).LE. HTMAX) GOTO 45
    IERR=4
    RETURN
45 IF (VEL(MNUM).GE. SPDMIN) GOTO 50
    IERR=5
    RETURN
50 IF (MNUM.EQ. 2) RETURN
IF (ACLIFT.LE. GMAX) GO TO 51
    IERR=1
    RETURN

```

```

C
C
C COMPUTE DRAG AND LIFT FORCES

```

```

51 RH0=0.0256*EXP(-0.103*(Z(MNUM)/1000.))
IF (Z(MNUM).GE. 10670.) GO TO 52
    RH0=0.00793*EXP(-0.156*(Z(MNUM)-10670.)/1000.)
52 CL=2*ACLIFT/(RH0*(VEL(MNUM)**2)/WL)
IF (CL.LE. CLMAX) GO TO 53
    IERR=17
    RETURN
53 DW=((CDO+CDK*CL*CL)/CL)*ACLIFT
TW=TMD+DW
IF ((TW.LE. TMAX) .AND. (TW.GE. 0.)) GOTO 55
    IF (TW.GT. TMAX) IERR=18
    DW={RH0*(VEL(MNUM)**2)/WL}* (0.05+CDO+CDK*(CL**2))/2.
    IF (TMD+DW.LE. 0) IERR=1
    IF (IERR.NE. 0) RETURN
55 IF (MNUM.NE. MBR) RETURN

```

```

C
C
C EVALUATE BOMBING RUN

```

```

DT=T(MNUM)-T(MNUM-1)
TGTHDG=ATAN2(DY,DX)*57.29578
IF (TGTHDG.LT. 0.) TGTHDG=TGTHDG+360.
DX=X(MNUM)-X(MNUM-1)
DY=Y(MNUM)-Y(MNUM-1)
ACHDG=ATAN2(DY,DX)*57.29578
IF (ACHDG.LT. 0.) ACHDG=ACHDG+360.
HDGLMT=AES(TGTHDG-ACHDG)
IF (POPALT.GE.1000.) GOTO 58
    IERR=6
    RETURN
58 IF (Z(MNUM).GE.100.) GOTO 60
    IERR=7
    RETURN
60 IF (Z(MNUM).LE. 2000.) GOTO 65
    IERR=8
    RETURN
65 IF (DIST.LE. 2000.) GOTO 75

```


APPENDIX C

KBPIP MODIFIED MODULES

```
COMMON /LOC/ X1,Y1,Z1,V1,LTR,LTR2
COMMON /MN/ IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
COMMON /PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
COMMON /PAR1/XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
COMMON /PAR2/T(200),XGUN(7),YGUN(7),ZGUN(7),XSAM,YSAM,ZSAM,GR(7)
COMMON /OPT/IGUN,IPNCH,TEXT,ISAM,IMP,KER
COMMON /TAR/TARGX,TARGY
```

CALL ERRSET SUPPRESSES ANY POSSIBLE UNDERFLOW PROBLEMS THAT MAY
RESULT FROM MANIPULATION OF SCENARIO PARAMETERS.

```
CALL ERRSET (208,50,-1,1,1)
MER=0
MNUM=0
CALL BEGIN
CALL SCENE(0,0)
IF (IGUN .NE. 1) GOTO 250
```

ACCEPT USER INPUT OF GUN SITES

```
REWIND 10
CALL ERRMK(16)
LTR=0
LTR2=0
DO 212 I=1,6
  IERR=0
  CALL XYZIN(0)
  XGUN(I)=X1
  YGUN(I)=Y1
  ZGUN(I)=Z1
  IF(Z1.GT.1000.) STOP
  IF(I.GE.5) CALL GUNCHK(X1,Y1,IERR)
  IF(IERR.EQ.0) GOTO 212
  CALL ERRMK(IERR)
  GOTO 211
```

```
WRITE (10,501) X1,Y1,Z1
CALL ERRMK(14)
CALL XYZIN(0)
XSAM=X1
YSAM=Y1
ZSAM=Z1
IF(Z1.GT.1000.) STOP
CALL SAMCHK(X1,IERR)
IF(IERR.NE.0) GOTO 220
WRITE(10,501) X1,Y1,Z1
```

IGUN=2==> READ GUN SITES FROM DATA FILE

```
IF (IGUN .NE. 0) GO TO 260
DO 252 I=1,6
  READ(10,501) XGUN(I),YGUN(I),ZGUN(I)
  READ(10,501) XSAM,YSAM,ZSAM
```



```

C      DRAW GUN SITES AND ENGAGEMENT CIRCLES
C
260 DO 265 I=1,7
265 CALL GUNLOC (XGUN(I),YGUN(I),GR(I))
    CALL GUNLOC (XSAM,YSAM,150.)
C
C      THIS SECTION ACCEPTS THE FLIGHT MILESTONES
C
    PTFAC=3.28084
    DGFAC=57.29578
    CALL ERRMK(15)
    REWIND 11
300 MNUM=MNUM+1
C
C      READ MILESTONES FROM THE DISK
C
302 IF (IMP.NE.0) GOTO 305
    READ(11,625) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),LTR
    IF (X(MNUM).LE.0.1) GOTO 302
    IF (LTR.EQ.1) IMP=1
    LTR2=LTR
    GOTO 308
C
C      READ MILESTONES FROM THE TERMINAL
C
305 CALL XYZIN(1)
    X(MNUM)=X1
    Y(MNUM)=Y1
    Z(MNUM)=Z1
C
C      COMPUTE FLIGHT PARAMETERS AND CHECK FOR ERRORS OR USER COMMANDS
C
    VEL(MNUM)=V1
308 IF (MNUM.EQ.1 .AND. LTR.EQ.83) STOP
    IF (MNUM.EQ.1) GOTO 300
    IERR=0
    CALL VALSET(IERR)
    IF ((LTR.EQ.82) .OR. (LTR2.EQ.82)) .AND. (IMP.NE.0)) GOTO 350
    IF ((LTR.EQ.66) .OR. (LTR2.EQ.66)) MBR=MNUM
    IF ((LTR.EQ.83) .OR. (LTR2.EQ.83)) GOTO 370
    IF (IERR.EQ.0) CALL ERRCHK(IERR)
    IF ((IERR.NE.0) .AND. (IERR.NE.12)) GOTO 310
    CALL ERRMK(IERR)
    IF (IMP.NE.0) GO TO 305
    CALL ERRMK(20)
    READ(5,*) ICOR
    IF (ICOR.EQ.1) GO TO 305
C MILESTONE ACCEPTED, RESTART THE SEQUENCE
310 CALL PTHPLT
    IF (LTR.EQ.65 .OR. LTR2.EQ.65) CALL AIMPT(MNUM)
    GOTO 300
C
C      THIS SECTION RESETS THE DATA
C
350 CALL SCENE(1,ICT)
    REWIND 19
    LTR=0
    LTR2=0
    IF(MBR .GT. ICT) MBR=0
C
C      COMPLETE RESTART
C
    IF (ICT .GT. 1) GOTO 352
    MNUM=0
    GOTO 260
352 IF ((ICT .GE. MNUM) .AND. (MNUM .GE. 3)) ICT=MNUM-1
    DO 355 MNUM=2,ICT
355 CALL PTHPLT

```


CCC

370

CCC

371

372

CCCC

375

376

CCC

377

[illegible]

150


```

COMMON IEAUD,MINY,MAXY,MINX,MAXX
WRITE(6,600) MNUM

C
C   SAVE MILESTONE DATA IN TEMP DATA
C
WRITE(19,625) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),MBR
600  FORMAT(' MILESTONE ',I3,' ACCEPTED')
625  FORMAT(4F10.0,I3)
RETURN
END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   GUNLOC
C   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE GUNLOC(GX,GY,RAD)
COMMON IEAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
RETURN
END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   SUBROUTINE WIN
C   THIS ROUTINE DEFINES A WINDOW EXTENDING FROM LX TO MX ON
C   THE HORIZCNTAL AXIS, AND FROM MINY TO MAXY ON THE VERTICAL
C   AXIS. THE HORIZONTAL RANGE IS RX, THE VERTICAL RANGE IS RY
C   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE WIN(LX,MX,RX,RY,JMP)
COMMON IEAUD,MINY,MAXY
RETURN
END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   SPOT
C   THIS RETURNS AN X-Y PAIR AND TWO COMMAND VALUES L1&L2
C   THE ROUTINE CREATES A DOT AT POINT X,Y AND ASKS FOR
C   OPERATOR VERIFICATION (ASCII"N"=78)
C   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE SPOT(X,Y,L1,L2)

C
C   L1=0 READ X AND Y
C
IF (L1 .EQ. 1) GOTO 10
WRITE(6,600)
READ(5,*) X,Y
GOTO 20

C
C   L2=0 READ ALTITUDE, OTHERWISE READ ALTITUDE, VELOCITY, AND A CMD
C
10 IF (L2 .EQ. 0) GOTO 15
WRITE(6,610)
WRITE(6,620)
READ(5,*) Y,X,L1

C
C   CONVERT COMMAND TO PROPER FORMAT
C
IF (L1 .EQ. 1) L1=65
IF (L1 .EQ. 2) L1=66
IF (L1 .EQ. 3) L1=82
IF (L1 .EQ. 4) L1=83
GOTO 20
15 WRITE(6,630)
READ(5,*) Y
20 L2=L1
600 FORMAT(' ENTER X AND Y COORDINATES')
610 FORMAT(' ENTER ALTITUDE,VELOCITY, AND AN OPTION')
620 FORMAT(' OPTION: 0=NONE; 1=AIM; 2=BOMB; 3=RESET; 4=STOP')
630 FORMAT(' ENTER ALTITUDE')
RETURN
END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   SUBROUTINE SCENE
C

```



```

C      THIS ROUTINE DRAWS THE ATTACK MAP
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE SCENE(IRO,ICT)
COMMON /LOC/ X1,Y1,Z1,V1,LTR,LTR2
COMMON /PAR/ X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
COMMON /ERR/ MKERX(3),MKERY(15)
COMMON /MN/ IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1

C
C      ASK USER FOR RESET NUMBER
C
IF (IRO.EQ. 0) GOTO 10
WRITE(6,600)
READ(5,*) ICT

C
C      DISPLAY DATA FOR FINAL POINT OF THE RESET AS OPERATOR AID
C
WRITE(6,650) X(ICT),Y(ICT),Z(ICT),VEL(ICT)
10 RETURN
600 FORMAT(' AT WHICH MILESTONE DO YOU WISH TO RESTART')
650 FORMAT(' X,Y,Z,AND VELOCITY',4F9.1)
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      XYZIN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      SUBROUTINE XYZIN(IV)
COMMON /LOC/ X1,Y1,Z1,V1,LTR,LTR2
COMMON /PAR/ IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
CALL WIN(MINX,MAXX,18000.,12000.,1)

C
C      GET X,Y AND THE FIRST COMMAND
C
LTR=0
CALL SPOT(X1,Y1,LTR,LDM)
CALL WIN(MIN1,MAX1,350.,2200.,1)
LTR2=1
LDM=IV

C
C      GET VEL AND,IF IV=1,ALT AND THE SECOND COMMAND
C
CALL SPOT(V1,Z1,LTR2,LDM)

C
C      CONVERT LOWER CASE TO UPPER CASE
C
IF (LTR.GT. 96) LTR=LTR-32
IF (LTR2.GT. 96) LTR2=LTR2-32
RETURN
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      SUBROUTINE BEGIN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      SUBROUTINE BEGIN
COMMON /MN/ IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1,MAP
COMMON /OPT/ IGUN,IPNCH,IEXT,ISAM,IMP,KER

C
C      READ USER OPTIONS
C
50 WRITE(6,600)
READ(5,*) IGUN
WRITE(6,630)
READ(5,*) IMP
WRITE(6,635)
READ(5,*) KER
WRITE(6,640)
READ(5,*) KON
IF (KON.NE. 0) CALL CONCHG
600 FORMAT(' GUNS: 0=DISK FILE; 1=TERMINAL; 2=PRESET ')
630 FORMAT(' MILESTONE INPUT: 0=DISK FILE; 1=TERMINAL')
635 FORMAT(' ERROR CHECKING: 0=NO CHECKING; 1=CHECK FOR ERRORS')

```


[illegible]

155


```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C          SUBROUTINE  ERRCHK                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE  ERRCHK(IERR)
      COMMON/PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
      COMMON/PAR1/XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
      COMMON/PAR2/T(200)
      COMMON/TAR/TARGX,TARGY
      COMMON/PAR3/TMD,ACLIFT,CLMAX,WL,TMAX,CDO,CDK

```



```
COMMON/ PAR4/ APPMAX, HTMIN, HTMAX, SPDMIN, GMAX, POPMIN
DATA TMSAV/ -1./
DX=TARGX-X(MNUM)
DY=TARGY-Y(MNUM)
DIST=SQRT(DX**2 + DY**2)
```

C
C
C INITIALIZE POPALT AND TMSAV

```
IF ((MNUM.NE.2) .AND. (TMSAV.NE.-1.)) GO TO 15
POPALT=0.
TMSAV=TMAX
15 IF (MBR.EQ.0 .AND. TMAX.NE.TMSAV) TMAX=TMSAV
CALL ERRMK(22)
IF (MER.GE.1) GOTO 30
IF ((DIST.LT. POPMIN).OR. (Z(MNUM).LT. APPMAX)) GOTO 30
IERR=4
RETURN
30 IF (DIST.GT. POPMIN .AND. Z(MNUM).GT. POPALT) POPALT=Z(MNUM)
35 IF (Z(MNUM).GE. HTMIN) GOTO 40
IERR=3
RETURN
40 IF (Z(MNUM).LE. HTMAX) GOTO 45
IERR=4
RETURN
45 IF (VEL(MNUM).GE. SPDMIN) GOTO 50
IERR=5
RETURN
50 IF (MNUM.EQ.2) RETURN
IF (ACLIFT.LE. GMAX) GO TO 51
IERR=1
RETURN
```

C
C
C COMPUTE DRAG AND LIFT FORCES

```
51 RH0=0.0256*EXP(-0.103*(Z(MNUM)/1000.))
IF (Z(MNUM).GE. 10670.) GO TO 52
RH0=0.00793*EXP(-0.156*(Z(MNUM)-10670.)/1000.)
52 CL=ACLIFT/(RH0*(VEL(MNUM)**2)/WL)
IF (CL.LE. CLMAX) GO TO 53
IERR=17
RETURN
53 DW=((CDO+CDK*CL*CL)/CL)*ACLIFT
TW=TMD+DW
IF ((TW.LE. TMAX) .AND. (TW.GE.0.)) GO TO 55
IF (TW.GT. TMAX) IERR=18
DW=(RH0*(VEL(MNUM)**2)/WL)*(0.05+CDO+CDK*(CL**2))/2.
IF (TMD+DW.LE.0) IERR=1
IF (IERR.NE.0) RETURN
55 IF (MNUM.NE. MBR) RETURN
```

C
C
C EVALUATE BOMBING RUN

```
DT=T(MNUM)-T(MNUM-1)
TGTHDG=ATAN2(DY,DX)*57.29578
IF (TGTHDG.LT.0.) TGTHDG=TGTHDG+360.
DX=X(MNUM)-X(MNUM-1)
DY=Y(MNUM)-Y(MNUM-1)
ACHDG=ATAN2(DY,DX)*57.29578
IF (ACHDG.LT.0.) ACHDG=ACHDG+360.
HDGLMT=ABS(TGTHDG-ACHDG)
IF (POPALT.GE.1000.) GOTO 58
IERR=6
RETURN
58 IF (Z(MNUM).GE.1000.) GOTO 60
IERR=7
RETURN
60 IF (Z(MNUM).LE.2000.) GOTO 65
IERR=8
RETURN
65 IF (DIST.LE.2000.) GOTO 75
IERR=9
RETURN
```


FILE: T1 FORTRAN A NAVAL POSTGRADUATE SCHOOL

```

75      IF (HDGLMT .LE. 5.) GOTO 80
          IERR=10
          RETURN
80      IF (DT .GE. 2.33) GOTO 85
          IERR=11
85      TMAX=1.2*TMAX
          RETURN
      END

```

[illegible]

APPENDIX D

IBMPIP MODIFIED MODULES

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C          SUBROUTINE PTHPLT                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PTHPLT
      COMMON/PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
      COMMON/PAR1/XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
      COMMON/MN/IBAUD,MINY,MAXY,MINX,MAXX
      REAL*4 TX1,TY1,TX2,TY2
      LOGICAL*1 CHARS(12)
      TX1=X(MNUM-1)
      TX2=X(MNUM)
      TY1=Y(MNUM-1)
      TY2=Y(MNUM)

C      IDENTIFY THE MAP, DRAW AND LABEL THE FLIGHT LEG
C
      CALL WIN(MINX,MAXX,18000.,12000.,0)
      CALL GBMCVE(TX1,TY1)
      CALL GBDRAW(TX2,TY2)
      IF (MNUM.GT.99) IW=3
      IF ((MNUM.GT.9) .AND. (MNUM.LT.100)) IW= 2
      IF (MNUM.LE. 9) IW=1
      CALL GAXITC(1,MNUM,IW,CHARS)
      CALL GBCHAR(TX2,TY2,2.,0.,IW,CHARS)
      CALL GSFRCE

C      SAVE THE PCINT IN 'TEMP DATA'
C
      WRITE(19,625) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),MBR
625  FORMAT(4F10.0,I3)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C          GUNLOC                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GUNLOC(GX,GY,RAD)
      COMMON/MN/IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
      REAL*4 X,Y,DX,DY
      X=GX
      Y=GY

C      IDENTIFY THE MAP AND PLACE A '+' AT THE LOCATION
C
      CALL WIN(MINX,MAXX,18000.,12000.,0)
      CALL GBMCVE(X-75.,Y)
      CALL GBDRAW(X+75.,Y)
      CALL GBMCVE(X,Y+75.)
      CALL GBDRAW(X,Y-75.)

```



```
C DETERMINE NUMBER OF STEPS TO BE USED
C ISTEP=3
C IF (RAD.LT. 1600.) ISTEP=6
C DRAW A CIRCLE AROUND THE SITE
C DO 10 I=3,180,ISTEP
C   ANGLE=I*.034907
C   DX=RAD*COS(ANGLE)+X
C   DY=RAD*SIN(ANGLE)+Y
C   CALL GBMOVE(DX+10.,DY)
10 CALL GBDFAW(DX-10.,DY)
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SPOT
C THIS RETURNS AN X-Y PAIR AND TWO COMMAND VALUES L1&L2
C THE ROUTINE CREATES A DOT AT POINT X,Y AND ASKS FOR
C OPERATOR VERIFICATION (ASCII"N"=78)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE SPOT(X,Y,L1,L2)
REAL*4 TX,TY,DOT
INTEGER*4 L,M,KEY
C READ THE CURSOR
C 100 CALL GBRXYC(M,L,KEY,TX,TY)
C   DOT=40.
C   IF(TX .LT.350.) DOT=9.
C MARK THE SPOT
C CALL GBGLOT(DOT,TX,TY)
C X=TX
C Y=TY
C DOES THE USER ACCEPT THE SPOT?
C CALL GBRXYC(M,L,KEY,TX,TY)
C IF (M .EQ. 0) GOTO 100
C CONVERT THE COMMAND TO THE PROPER FORMAT
C L1=0
C IF (M .NE. 1) GOTO 110
C IF (L .EQ. 1) L1=65
C IF (L .EQ. 2) L1=66
C IF (L .EQ. 3) L1=82
C IF (L .EQ. 4) L1=83
110 L2=L1
RETURN
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SCENE
C THIS ROUTINE DRAWS THE ATTACK MAP
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE SCENE(IRO,ICT)
COMMON/ERR/MKEX(3),MKERY(18)
COMMON/MN/IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
COMMON/PAR/X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
COMMON/TAR/TARGX,TARGY
REAL*4 GX,GY
LOGICAL*1 CHARS(20)
ICT=0
C ERASE THE SCREEN
```


161


```

225 CALL GBDRAW(I*50.,0.)
C MARK RESET POINT VELOCITY AND ALTITUDE AS OPERATOR AID
C IF (IRQ.NE.0 .AND. ICT .GT.1) CALL GBGDOT(9.,VEL(ICT),Z(ICT))
C MAXY=MINY-15
C MINY=MINY-200
C CALL WIN(MIN1-250,MAX1+650,195.,185.,0)
C CALL GBCHAR(10.,110.,2.0,0.,19,'ALT MARKS IN 1000"S')
C CALL GBCHAR(10.,60.,2.0,0.,29,'VELOCITY IN 50"S ALONG X AXIS')
C CALL GSFRCE
C MAXY=I2
C MINY=I1
501 FORMAT(2F10.2)
630 FORMAT(' AT WHICH POINT DO YOU WANT TO RESTART')
635 FORMAT(' VILLAGE: 0=NOT DRAWN; 1=VILLAGE DRAWN')
650 FORMAT(' X,Y,Z,AND VELOCITY',4F9.1)
RETURN
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SUBROUTINE WIN C
C THIS ROUTINE DEFINES A WINDOW EXTENDING FROM LX TO MX ON C
C THE HORIZONTAL AXIS, AND FROM MINY TO MAXY ON THE VERTICAL C
C AXIS. THE HORIZONTAL RANGE IS RX, THE VERTICAL RANGE IS RY C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE WIN(LX,MX,RX,RY,JMP)
COMMON/MN/IBAUD,MINY,MAXY
REAL*4 WIN1(4)
INTEGER*4 V1(4)
DATA WIN1(1)/0./,WIN1(2)/0./

C DEFINE THE WINDOW
C
C V1(2)=MINY
C V1(4)=MAXY
C WIN1(3)=RX
C WIN1(4)=RY
C V1(1)=LX
C V1(3)=MX
C CALL GASVIE(V1,0)
C CALL GASWIN(WIN1,0)
C IF (JMP .EQ. 0) GOTO 11
C FLASH THE DEFINED WINDOW
C
C ICT=4+50*IBAUD
C DO 10 I=1,ICT
C CALL GBMOVE(0.,0.)
C CALL GBDRAW(WIN1(1),WIN1(4))
C CALL GBDRAW(WIN1(3),WIN1(4))
C CALL GBDRAW(WIN1(3),WIN1(2))
10 CALL GBDRAW(WIN1(1),WIN1(2))
11 RETURN
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SUBROUTINE BEGIN C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE BEGIN
COMMON/MN/IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1
COMMON/OPT/IGUN,IPNCH,IEXT,ISAM,IMP,KER
C READ USER OPTIONS
C
C 50 WRITE(6,600)
C READ(5,*) IGUN
C WRITE(6,630)
C READ(5,*) IMP
C WRITE(6,635)
C READ(5,*) KER

```



```

WRITE (6,625)
READ (5,*) IBAUD
WRITE (6,640)
READ (5,*) KON
IF (KON .NE. 0) CALL CONCHG
600  FORMAT(' GUNS: 0=DISK FILE; 1=TERMINAL; 2=PRESET ')
625  FORMAT(' BAUD RATE: 0=300 BAUD; 1=1200 BAUD ')
630  FORMAT(' MILESTONE INPUT: 0=DISK FILE; 1=TERMINAL ')
635  FORMAT(' ERROR CHECKING: 0=NO CHECKING; 1=CHECK FOR ERRORS ')
640  FORMAT(' FLIGHT & GAME PARAMETERS: 0=DEFAULT; 1=USER INPUT ')

C
C
C  INITIALIZE THE GRAPHICS AND CLEAR SCREEN

MAXX=3200
MINX=500
MAX1=MAXX+225
MIN1=MAXX+30
MAXY=2500
MINY=700

C
C
C  COMPUTE THE SCREEN WINDOW COORDINATES

CALL DSINIT
CALL GSERSE
RETURN
END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC                                     ELFIN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE ELFIN (LAST)
COMMON/OPT/IGUN,IPNCH,IEXT,ISAM,IMP,KER
COMMON/MN/ IBAUD,MINY,MAXY,MINX,MAXX,MIN1,MAX1

C
C
C  CLOSE GRAPHICS ROUTINES

CALL DSTERM

C
C
C  OLD (LAST=0) OR NEW (LAST=1) FLIGHT PATH?

IF (LAST .EQ. 0) GOTO 20

C
C
C  REQUEST USER OUTPUT OPTIONS

15  WRITE (6,595)
    READ (5,*) LAST
    IF (LAST.GE.2 .OR. LAST.LT.0) GO TO 15
    LAST=1 + 82*LAST
20  WRITE (6,610)
    WRITE (6,620)
    READ (5,*) IPNCH

C
C
C  IF MILESTONES CAME FROM DISK, RETURN END OF FILE MARKER

IF (LAST .EQ. 0) LAST=83
IF (IPNCH .EQ. 0) RETURN
IF (IPNCH.EQ.2) GOTO 100
    WRITE (6,630)
    READ (5,*) IEXT
    IF (IPNCH .EQ. 1) RETURN
100  WRITE (6,640)
    READ (5,*) ISAM
    IF ((ISAM .LT. 1) .OR. (ISAM .GT. 7)) GOTO 100
    RETURN
590  FORMAT(' +HIT ANY KEY FOR OUTPUT OPTION SELECTION ')
595  FORMAT(' ARE YOU FINISHED WITH THIS FLIGHTPATH: 0=NO; 1=YES ')
600  FORMAT(' SCREEN OUTPUT: 0=NO OUTPUT; 1=OUTPUT DESIRED ')
610  FORMAT(' OUTPUT FILE: 0=NO OUTPUT; 1=POO1 FILE ONLY ')
620  FORMAT('                2=MICE FILE ONLY; 3=POO1 & MICE FILES ')
630  FORMAT(' EXTENDED OUTPUT: 0=NOT WANTED; 1=EXTENDED OUTPUT ')
640  FORMAT(' MISSILE TYPE BETWEEN 1 AND 7 ')

```


[illegible]


```

      READ(5,*) APPMAX
      GO TO 30
10    WRITE(6,210)
      READ(5,*) TMAX
      GO TO 30
11    WRITE(6,211)
      READ(5,*) CDO
      GO TO 30
12    WRITE(6,212)
      READ(5,*) CDK
      GO TO 30
13    WRITE(6,213)
      READ(5,*) VMAX1
      GO TO 30
14    WRITE(6,214)
      READ(5,*) VMAX2
      GO TO 30
15    WRITE(6,215)
      READ(5,*) TARGX,TARGY
      GO TO 30

C
C  READ NEW PARAMETER VALUES FROM THE DISK
C
16    REWIND 16
      READ (16,300) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      READ (16,300) POPMIN,APPMAX,TMAX,CDO,CDK,VMAX1
      READ (16,300) VMAX2,TARGX,TARGY
      GO TO 30

C
C  WRITE PARAMETER VALUES TO THE TERMINAL
C
17    WRITE(6,250)
      WRITE(6,300) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE(6,255)
      WRITE(6,300) POPMIN,APPMAX,TMAX,CDO,CDK,VMAX1
      WRITE(6,260)
      WRITE(6,300) VMAX2,TARGX,TARGY
      GO TO 30

C
C  SAVE PARAMETER VALUES ON DISK AND RETURN TO CALLING ROUTINE
C
200   REWIND 16
      WRITE(16,300) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE(16,300) POPMIN,APPMAX,TMAX,CDO,CDK,VMAX1
      WRITE(16,300) VMAX2,TARGX,TARGY

RETURN
100   FORMAT(' ')
105   FORMAT(' SELECT PARAMETER TO BE CHANGED: 1=NO MORE CHANGES')
110   FORMAT(' 2=MAX LIFT COEFF; 3=WING LOADING; 4=STALL SPEED')
115   FORMAT(' 5=MAX G FORCE; 6=MIN ALT; 7=MAX ALT(<2200 MTR)')
120   FORMAT(' 8=DISTANCE TO TARGET BEFORE POPUP ALLOWED')
125   FORMAT(' 9=MAX APPROACH ALT; 10=MAX T/W RATIO; 11=DRAG COEF. W/O
$ LIFT')
130   FORMAT(' 12=LIFT DRAG CONSTANT; 13=MAX SPD WITH BOMB')
135   FORMAT(' 14=MAX SPD W/O BOMB; 15=TARGET COORDINATES')
140   FORMAT(' 16=READ PARAMETER FILE; 17=LIST PARAMETERS')
202   FORMAT(' ENTER MAX LIFT COEFFICIENT')
203   FORMAT(' ENTER WING LOADING')
204   FORMAT(' ENTER STALL SPEED')
205   FORMAT(' ENTER MAX G FORCE ALLOWED')
206   FORMAT(' ENTER MINIMUM ALTITUDE')
207   FORMAT(' ENTER MAXIMUM ALTITUDE (LESS THAN 2000 METERS)')
208   FORMAT(' ENTER MINIMUM DISTANCE TO TARGET BEFORE POP-UP ALLOWED')
209   FORMAT(' ENTER MAXIMUM APPROACH ALTITUDE')
210   FORMAT(' ENTER MAX THRUST TO WEIGHT ALLOWED')
211   FORMAT(' ENTER DRAG COEFFICIENT FOR ZERO LIFT')
212   FORMAT(' ENTER LIFT DRAG CONSTANT')
213   FORMAT(' ENTER MAX SPEED CARRYING A BOMB')
214   FORMAT(' ENTER MAX SPEED AFTER BOMB IS RELEASED')
215   FORMAT(' ENTER TARGET X AND Y COORDINATES')
250   FORMAT(' MAX CL WING LD STALL SPD MAX G MIN.HT. MAX.HT.')
255   FORMAT(' POP-UP DIST APPR.HT. MAX THRUST CDO DRAG CONSTANT')

```


FILE: T2 FORTRAN A NAVAL POSTGRADUATE SCHOOL

```

6 10  FORMAT(' HDG>5 DEG TO TGT')
6 11  FORMAT(' FINAL RUN<2.33 SEC')
6 12  FORMAT(' NO HORIZONTAL MOTION')
6 13  FORMAT(' TOO CLOSE TO TGT')
6 14  FORMAT(' ENTER MISSILE LOCATION')
6 15  FORMAT(' ENTER MILESTONES')
6 16  FORMAT(' ENTER GUN LOCATIONS')
6 17  FORMAT(' MAX LIFT EXCEEDED')
6 18  FORMAT(' MAX THRUST EXCEEDED')
6 19  FORMAT(' X COORDINATE LESS THAN 6000')
6 20  FORMAT(' DO YOU WANT TO FIX THE ERROR:0=NO,USE THE POINT;1=YES')
6 50  FORMAT(' X,Y,Z,AND VELOCITY',4F9.1)
      END

```

000000000000
E-E-E-E-E-E-E-E-E-E

APPENDIX E

SCENE PROGRAM LISTING

```

DIMENSION LIMX (4) , LIMY (4)
CALL INIT
50  WRITE (6,600)
    READ (5,*) II
    ONE=-0.5
    TWO=-2.0
    ZERO=0.0
    MAXX=KIN (7.5)
    MINX=KIN (1.5)
    MAXY=KIN (0.375)
    MINY=KIN (4.375)
    MAXY=777-MAXY
    MINY=777-MINY
    READ (5,*) QT
    CALL NEWFAG
    CALL MOVABS (MINX, MINY)
    CALL DRWABS (MINX, MAXY)
    CALL DRWABS (MAXX, MAXY)
    CALL DRWABS (MAXX, MINY)
    CALL DRWABS (MINX, MINY)
    CALL DWINDO (0., 18000., 0., 12000.)
    CALL TWINDO (MINX, MAXX, MINY, MAXY)
    IF (II.GT.0) GOTO 300
    READ (9,501) X,Y
200  READ (9,501) X,Y
    CALL MOVEA (X,Y)
201  READ (9,501) X,Y
    IF (X.LT.-1) GOTO 300
    IF (X.LT. 0) GOTO 200
    CALL DRAWA (X,Y)
    GOTO 201
300  WRITE (9,501) ONE,ZERO
    CALL VCURSR (LTR,X,Y)
    CALL POINTA (X,Y)
    WRITE (9,501) X,Y
301  CALL VCURSR (LTR,X,Y)
    CALL DRAWA (X,Y)
    WRITE (9,501) X,Y
    IF (LTR.EQ.77) GOTO 300
    IF (LTR.NE.83) GOTO 301
    WRITE (9,501) TWC,ZERO
    CALL FIN
500  FORMAT (2I7)
501  FORMAT (2F10.2)
600  FORMAT (' ENTER POSITIVE INTEGER FOR RESTART')
STOP
END

```


LIST OF REFERENCES

1. Department of Defense Military Standard MIL-STD-2069, Requirements for Aircraft Nonnuclear Survivability, 24 August 1981.
2. Joint Technical Coordinating Group for Munitions Effectiveness, Technical Note 4565-16-73, Antiaircraft Artillery Simulation Computer Program--AFATL Program P001, Vols I and II, by J. Severson and T. McMurchie, September 1973.
3. Joint Technical Coordinating Group for Aircraft Survivability Report 78-S-001, Surface-To-Air Missile Modal--MICE II (User's Manual), The Vaught Corporation, April 1980.

BIBLIOGRAPHY

Maxwell, George Gary, The Development of Two Class Problems for a Course in Aircraft Combat survivability: A Study of Aircraft Attrition in a Hostile Antaircraft Artillery Environment and A Survivability Design and Campaign Analysis fo a Combat Aircraft, Master's Thesis, Naval Postgraduate School, Monterey, California, 1978.

TEKTRONIX, PLOT 10 Terminal Control System User's Manual, Beaverton, Oregon, June 1979.

INITIAL DISTRIBUTION LIST

	No. of Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 39340	2
3. Combat Data Information Center AFWAL/FIES/CDIC Wright-Patterson AFB, Ohio 45433	1
4. Mr. Martin Lentz ASD/ENFTV Wright-Patterson AFB, Ohio 45433	1
5. Professor R. E. Ball Code 67Bp Department of Aeronautics Naval Postgraduate School Monterey, California 39340	2
6. LCDR R. Stillwell, Code 52SB Naval Postgraduate School Monterey, California 39340	1
7. LT Eric Johns 5591 Soledad Mountain Road La Jolla, California 92037	1

Thesis
J5493 Johns
c.1

198098
Creation of a
transportable in-
teractive user in-
terface for improved
AAA simulation com-
puter program (P001).

Thesis
J5493 Johns
c.1

198098
Creation of a
transportable in-
teractive user in-
terface for improved
AAA simulation com-
puter program (P001).

thesJ5493

Creation of a transportable interactive



3 2768 002 10776 5

DUDLEY KNOX LIBRARY